UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

IMPLANTAÇÃO DE UMA PLATAFORMA INTEGRADORA DE TRABALHO E DESENVOLVIMENTO PARA A FCUL

José Romana Baptista Coelho

MESTRADO EM ENGENHARIA INFORMÁTICA
Sistemas de Informação

2010

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

IMPLANTAÇÃO DE UMA PLATAFORMA INTEGRADORA DE TRABALHO E DESENVOLVIMENTO PARA A FCUL

José Romana Baptista Coelho

PROJECTO

Trabalho orientado pelo Prof. Doutor Carlos Alberto Pacheco dos Anjos Duarte

e co-orientado por Pedro Miguel Gomes Silva Rosa

MESTRADO EM ENGENHARIA INFORMÁTICA
Sistemas de Informação

2010

Agradecimentos

Gostava de agradecer a todos os que tornaram possível este projecto, e nesse grupo relativamente grande de gente, gostava de agradecer especialmente ao Prof. Doutor. Carlos Duarte, ao Rui Nunes e ao Pedro Rosa principais envolvidos e coordenadores de todo o meu trabalho.

Gostava também de deixar um agradecimento especial ao Ricardo Batista, principal suporte na última fase deste trabalho e principal impulsionador de todos os testes efectuados e com interesse directo para este projecto. Sem a tua ajuda, teria sido impossível finalizar este projecto, e por isso te agradeço profundamente.

Aos restantes elementos do Centro de Informática, que conviveram comigo durante a maior parte do tempo deste projecto, o meu agradecimento pela forma exemplar como me trataram, agradecendo com especial carinho e amizade a Susana Pereira, João Soares e Paulo Bastos.

Agradeço a quem em primeiro lugar me possibilitou respirar, viver e todas as coisas que o mundo tem para oferecer. Pai e Mãe, são os meus modelos em vida, trabalho, amor, amizade e educação e os responsáveis por todas as qualidades que eu possa ter. Obrigado.

Agradeço também aos meus irmãos Paulo, Jeni e Pedro, todo o apoio e amizade incondicional que me deram. Mais que irmãos, são os melhores amigos que um irmão mais novo podia ter.

Por fim, mas nunca de forma redutora, agradeço aos que me acompanham todos os dias nas manias, nos gostos, e principalmente nos incontáveis defeitos, Filipa, Luís, Miana, Filipe, João, Pedro e Ruben obrigado do fundo do coração.

Para a minha família e amigos.

Resumo

O funcionamento de um sistema de ensino nos dias que correm, abrange mais que as próprias aulas e matérias, sendo indispensável aos alunos um acesso facilitado a sistemas de gestão e a aplicações que lhes permitam pagar propinas, inscreverem-se em exames, acederem a notas, etc. A própria Faculdade deixou de ser apenas um espaço onde se aprende, para passar a ser ela própria, uma espécie de fornecedora de serviços para que não falte nada aos seus alunos, docentes e funcionários.

Neste contexto, a adopção de uma metodologia de desenvolvimento e de ferramentas tecnológicas que facilitem o desenvolvimento e a gestão de serviços, por parte do Centro de Informática da Faculdade de Ciências da Universidade de Lisboa, é essencial para que seja garantida uma eficiente resposta às necessidades de toda a Faculdade e meio envolvente.

Este projecto procura a implementação e estruturação de uma metodologia ágil de desenvolvimento, definindo todos os seus processos, bem como todas as reponsabilidades e papéis dos diversos intervenientes e as ferramentas a utilizar para dar resposta ao desenvolvimento de aplicações e serviços por parte da FCUL.

Palavras-chave: Metologia, Conteúdos, CMS, Ágil, Desenvolvimento, Serviços

Abstract

The operation of an education system these days, covers more than their own lessons and materials, being indispensable to students easier access to management systems and applications that enable them to pay fees, enroll in examinations, access to notes, etc.. The University itself is no longer just a place where one learns to become itself a kind of service provider to not miss anything to their students, faculty and staff.

In this context, the adoption of a development methodology and technology tools that facilitate the development and management services by the Centro de Informática of University of Lisbon, it is essential to be guaranteed an efficient response to the needs of throughout the Faculty and the environment.

This project aims at the implementation and structuring of an agile development methodology, defining all its processes, and all share responsibilities and roles of various actors and the tools used to address the development of applications and services by the FCUL.

Keywords: Methodology, Content, CMS, Agile, Development, Services

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Enquadramento

1.1.1 Instituição de acolhimento:

Sobre a FCUL

A Faculdade de Ciências da Universidade de Lisboa - FCUL, é uma das unidades orgânicas que integram a Universidade de Lisboa. Actualmente, a FCUL, ocupa onze edifícios com uma área total de 77 492 m 2, no campus do Campo Grande.

A FCUL tem à disposição do aluno um conjunto de modernas infra-estruturas com o objectivo de lhe dar todas as condições para um ensino de excelência. Integram-se ainda no campus da FCUL o IBEB (Instituto de Biofísica e Engenharia Biomédica), o Instituto de Oceanografia e o ICAT (Instituto de Ciência Aplicada e Tecnologia). Fora do campus, o Laboratório Marítimo da Guia (Guincho), o Observatório Astronómico de Lisboa e a Estação de Campo do CBA (Grândola).

A Faculdade de Ciências de Lisboa está estruturada em 10 Departamentos e 1 Secção Autónoma que conduzem o ensino e a investigação nos respectivos domínios científicos: Biologia Animal, Biologia Vegetal, Educação, Estatística e Investigação Operacional, Engenharia Geográfica, Geofísica e Energia, Física, Geologia, Informática, Matemática, Química e Bioquímica e História e Filosofia das Ciências.

Tem mais de duas dezenas de Unidades de I&D financiadas e avaliadas pela Fundação para a Ciência e Tecnologia, e que se dedicam a múltiplas áreas das ciências e das tecnologias, com relevo para a Matemática, a Física, a Química, a Biologia, a Geologia, a Estatística e Investigação Operacional, a Informática, a Engenharia Geográfica e Geofísica, a História e Filosofia das Ciências e a Educação.

As Unidades de I&D acolhidas na FCUL têm obtido classificações elevadas nos painéis internacionais de avaliação, dando continuidade à tradição de qualidade científica da Escola e de afirmação crescente no plano internacional. O seu financiamento é assegurado pela FCT, a União Europeia ou por Investigação sob contrato com empresas e organismos públicos. Nelas desenvolvem actividade os docentes e os investigadores integrados nessas unidades, bem como os doutorandos e pós-doutorados oriundos de todo o mundo, que escolheram a FCUL para desenvolver actividade científica. Aqui estão instalados os equipamentos relevantes e as bibliotecas de investigação onde se vive o dia-a-dia da aventura da ciência.

A FCUL é um dos principais participantes nacionais nos Programas Europeus de Investigação e Desenvolvimento Tecnológico, parceira dos acordos de cooperação internacional estabelecidos por Portugal com Universidades dos EUA, promove a colaboração científica bilateral e multilateral, a ligação

entre investigação e indústria e o empreendorismo. Alguns destes projectos têm uma vertente tecnológica significativa, uma vez que as aulas são dadas em Vídeo-conferência através da Internet, dispondo ainda de laboratórios específicos para estes projectos.

A FCUL tinha, em 2008/2009, 5061 alunos, distribuídos pelos vários ciclos de ensino e 634 Funcionários, entre os quais 419 Docentes.

A dimensão desta Instituição, e a grande quantidade de informação que esta gere, exige a existência de um grupo coordenador de todos os recursos humanos bem como de todos os recursos informáticos. É nesse ponto que entra o Centro de Informática e todos os serviços e recursos que este oferece.

Sobre o CI

O Centro de Informática é a Unidade Orgânica da Faculdade de Ciências da Universidade de Lisboa responsável pela gestão da rede Internet da Faculdade de Ciências, bem como dos diversos serviços a ela associados, nomeadamente e-mail e páginas Web. A manutenção de alguns sistemas de informação, tais como as bases de dados relativas à gestão dos utilizadores (docentes, funcionários e alunos) e as correspondentes páginas amarelas, está também a cargo do CI.

O Centro de Informática tem ainda a seu cargo o apoio técnico a todos funcionários da Faculdade, através de um serviço de Help Desk. Contudo, a prestação de apoio especializado é realizada no âmbito de acordos de colaboração com as várias unidades orgânicas da FCUL. Este apoio inclui:

- Gestão e apoio técnico (instalação hardware e software)

- Gestão de laboratórios de alunos

- Aconselhamento e acompanhamento na aquisição de software e hardware

Outros serviços também prestados de uma forma regular, e igualmente no âmbito de acordos expressos, incluem:

- Apoio a conferências ou reuniões científicas

- Alojamento de equipamentos

O CI mantém também acordos com unidades da Universidade de Lisboa externas à FCUL, projectando a faculdade no exterior e permitindo uma melhor gestão dos seus recursos humanos.

A equipa que forma o CI é constituída por: um Coordenador Geral, Pedro Rosa; responsáveis das infra-estruturas de sistemas e de redes, respectivamente Paulo Bastos e Pedro Botas; uma equipa de gestão e desenvolvimento dos sistemas de informação composta por Rui Nunes, Rui Batista, João Soares, Ricardo Batista e a minha pessoa; uma Técnica Administrativa, Eugénia Graça; e por uma equipa de Operação constituída por Joel Fonseca, Susana Pereira, Diogo Albuquerque, David Silva, Tiago Felgueiras, Ricardo Costa, Rui Posse, André Pinto, Pedro Maia, João Lopes e Maiur Chhaganlal.

1.1.2 PEI na FCUL:

A realização do Projecto em Engenharia Informática instituído pelo Departamento de Informática da FCUL é, no âmbito do meu estágio, efectuado no Centro de Informática da mesma instituição, representando um proveito interno para a própria faculdade que pode assim tirar proveito directo da formação dada a um seu próprio instruendo. Esta "troca" interna tem na realidade todo o sentido, uma vez que representa também para mim uma oportunidade única de trabalho, facilitando a integração do estágio e possibilitando uma relação estreita entre a formação me que foi dada durante quatro anos de licenciatura e mestrado, e o estágio realizado como passo final para a conclusão do segundo.

As vantagens são portanto óbvias para todas as partes envolvidas neste projecto: a faculdade lucra duplamente com a formação de um aluno e com seu contributo directo no contínuo desenvolvimento de um órgão de gestão interna, e o estagiário lucra com uma facilidade maior de integração e uma passagem gradual de um ambiente académico para um ambiente profissional, sendo que também todo o processo de gestão do estágio é simplificado uma vez que apenas uma instituição está envolvida no processo não existindo vontades externas que lhe possam ser obstáculo.

Equipa em que o aluno foi enquadrado:

No Centro de Informática da Faculdade de Ciências existe um pequeno número de pessoas que procuram dar resposta a um grande número de serviços que procuram servir não só a própria faculdade, como também contribuir para a gestão de muitas instituições/grupos que rodeiam a Universidade em que a FCUL se insere. Tarefas predominantemente de gestão de aplicações e serviços académicos, bem como desenvolvimento de outros serviços, representam uma forma de dar resposta a um mundo académico em constante evolução tecnológica e de processos. Esse pequeno número de pessoas tem por isso de ser capaz de se "desdobrar" consecutivamente para dar resposta adequada a diferentes tipos de exigências.

Uma equipa que é essencialmente constituída por cerca de 15 pessoas, distribuídas por dois tipos de tarefas bem distintas, em que cerca de metade presta auxílio directo ao meio académico, através de operações de suporte relativamente simples, como gestão de acessos a serviços e aplicações, enquanto a outra metade situa-se no centro de toda a implementação e desenvolvimento dos próprios serviços, garantindo o correcto funcionamento de toda a faculdade de forma interna e com o meio envolvente.

Em todo o trabalho desenvolvido está, no entanto, ausente um processo bem definido no que diz respeito ao desenvolvimento de novos sistemas, e ao suporte eficaz dos produtos já existentes para toda a faculdade. Por essa mesma razão, e pelo facto de ser reconhecido por todos a necessidade de uma metodologia de desenvolvimento mais eficiente, urge a execução deste PEI no qual me enquadro, estando reunidas as condições ideais para a sua concretização. Com uma metodologia e respectivas formas de trabalho devidamente delineadas e definidas em todo o grupo de trabalho e com a definição clara dos papéis de cada trabalhador em cada projecto torna-se possível um crescimento para o futuro devidamente sustentado e uma gestão simplificada dos diferentes serviços fornecidos pelo CI.

Possibilitando todo este desenvolvimento a actual equipa encontra-se agora em expansão como nunca se encontrou antes, tendo praticamente duplicado a sua dimensão nos últimos tempos, integrando na sua constituição estagiários capazes de participar na implementação de novos sistemas de informação como resposta urgente às necessidades de evolução da instituição, e preparando a sua permanência após um período de aprendizagem relativamente rápido (possibilitado pela pequena dimensão da equipa).

Foi nesta pequena equipa que fui integrado a início num estágio independente deste projecto de Engenharia Informática, para desempenhar funções de gestão de permissões de acesso aos vários sistemas de dados existentes ao nível de toda a faculdade, bem como na simplificação de tarefas e em actividades de desenvolvimento de sistemas de informação (tanto existentes como novos serviços). Ao fim de alguns meses no desempenho destas funções encontrava-me à data de início do PEI, já bastante familiarizado com todos os processos de trabalho e com todos os sistemas e tipos de informação existentes por toda a FCUL-CI. Isto constituiu uma vantagem importante no sentido de tornar possível um projecto tão extenso e abrangente como o que foi realizado.

Para este projecto, contei inicialmente com a colaboração directa de uma pessoa – Rui Nunes – que tipicamente desempenhou (em conjunto com o Coordenador do PEI, Pedro Rosa) funções de coordenação do Projecto e de validação do trabalho desenvolvido pela minha pessoa. Com o avançar do projecto pude ainda contar com a colaboração directa de Ricardo Batista na implementação específica de funcionalidades em diversas ferramentas de desenvolvimento para constituição de casos de teste que procuram validar a utilização das diferentes ferramentas na metodologia de trabalho definida. Quanto aos restantes elementos que integram o Centro de Informática, tiveram um papel activo na experimentação e no fornecimento de opiniões e sugestões para a escolha da metodologia e das ferramentas que mais convinham à instituição, e que mais possibilidades apresentavam de ser implementadas de forma correcta e total.

Quanto ao meu papel no projecto foi claramente um papel primário (como é possível de verificar no capítulo de descrição do projecto), cabendo-me a mim a investigação de metodologias e construção de uma forma de desenvolvimento baseada em uma ou mais formas de trabalho já devidamente testadas e documentadas (com relatos de usos anteriores e identificação de todos os pontos fracos, pontos fortes e riscos). Essa especificação de uma metodologia ágil de desenvolvimento permitirá a toda a equipa de desenvolvimento adquirir práticas mais sistemáticas e sustentadas, com vista a um desenvolvimento mais eficaz e controlado, capaz de responder de forma mais imediata à implementação de novos serviços, oferecendo simplificação ao suporte dos mesmos.

O tamanho e o tipo de equipa, bem como o espaço aberto em que se processa todo o trabalho, levam a uma relação próxima entre os vários membros da equipa do Centro de Informática da FCUL, propiciando-me também por essa razão uma integração mais eficaz com as formas de trabalho e colegas.

Enquadramento do Projecto no mundo de Projectos do CI

Um factor a ter em conta em todo este projecto, é o facto de os limites do mesmo serem muito ténues e definidos de uma forma pouco rígida. Isso acontece uma vez que grandes etapas do projecto são conhecidas de uma forma expectante e não com base em certezas. Veja-se, por exemplo, a integração de diferentes ferramentas nas diversas fases da metodologia de desenvolvimento, que tanto pode ocorrer de uma forma tranquila desde o seu início, como pode exigir alterações não calculadas inicialmente, ou mesmo o abandono de certas ferramentas e o avanço de novas etapas de experimentação com ferramentas alternativas.

Para além desta implementação do "desconhecido a início", temos também o facto de este projecto implicar grandes alterações em toda a organização que a comporta e poder originar grande número de outros desenvolvimentos/implementações futuras. A tentação de passar além do projecto e entrar nesses projectos futuros é sempre grande e por isso terá de haver sempre um grande controlo para que isso não aconteça, principalmente através de uma definição exigente de objectivos e capacidade de não passar para além destes evitando uma dimensão irreal indesejada.

Sabemos assim, que todos os outros projectos desta instituição se terão que relacionar no futuro com o produto resultante deste PEI, e por isso lhe conferimos um aspecto central em todo o trabalho de desenvolvimento efectuado pela instituição. Temos no entanto em conta, que todo o normal funcionamento e implementação de serviços tem de continuar a ser feito de forma paralela, para que não seja afectado o funcionamento da instituição. Projectos paralelos terão naturalmente que ignorar (para já) especificações que possam ser definidas ao nível da definição de uma metodologia de desenvolvimento, para só depois no futuro voltarem a ser integrados segundo novos padrões que possam ter sido definidos.

1.2 Objectivos

1.2.1 Apresentação do problema a resolver:

Problema:

No Centro de Informática da FCUL há uma clara ausência de metodologias de trabalho, bem como de metodologias de desenvolvimento de serviços. É tarefa primordial deste projecto definir precisamente uma metodologia capaz de englobar todos os órgãos e conjunto de pessoas que integram o CI e especificar de uma forma exaustiva - mas sempre simples – um rumo de tarefas a seguir aquando da implementação de um qualquer novo sistema por parte desta unidade orgânica.

A interacção entre as várias pessoas que formam esta pequena equipa de desenvolvimento assume um papel central na adopção de uma nova forma de trabalho, sendo essencial uma comunicação eficaz entre todos os elementos, para que o desenvolvimento de novos projectos seja sempre uma tarefa colectiva. É do senso comum, que se a metodologia não se começar por adoptar às pessoas, também estas

não se mostrarão interessadas em mudar costumes e hábitos para abraçarem novos processos e uma nova metodologia. Sendo uma equipa de trabalho relativamente pequena e actualmente em expansão torna-se ainda mais imperativo que todos aceitem os novos processos e que haja uma coordenação ideal entre as tarefas desempenhadas por cada um. Só partindo dessa coordenação é possível alcançar a organização e por conseguinte o sucesso.

No presente, a aceitação de novos projectos e a implementação destes, parte de uma auto-suficiência por parte dos seus desenvolvedores. Não há uma ordem certa para se implementarem as várias tarefas que constituem um projecto, e as próprias tarefas não estão na realidade bem definidas. O projecto é encarado como um todo, projectado como um todo e desenvolvido como um todo, podendo levar ao seu rotundo insucesso (que por vezes se poderá traduzir apenas num alargamento quase infinito do seu prazo de conclusão).

Tendo em conta que o desenvolvimento de serviços informáticos e o trabalho relacionado com estes já existe há muitos anos na FCUL e que durante todos esses anos nunca houve uma documentação bem definida do trabalho desenvolvido ao nível informático – projectos, formas de gestão, novos serviços, etc. – reflecte-se que todo o desenvolvimento e comportamento presente do CI, vive graças a certas individualidades responsáveis pelo acompanhamento e realização de grande número de projectos realizados no passado. Podemos assim dizer, que existe no CI um conjunto de pessoas que juntas representam a informação representativa das características e do funcionamento de toda a FCUL, e que em vez de repositórios de informação, existem essencialmente "mentes de informação" com o conhecimento necessário para qualquer desenvolvimento da responsabilidade do CI. A integração de novas pessoas, tal como a implementação de novos projectos, passa necessariamente pelo debate de ideias em que essas mentes se encontram no centro e são responsáveis pela transferência de conhecimento para os restantes envolvidos.

Esta falta crítica de informação é não só uma barreira enorme à reformulação de processos de trabalho, como também é uma dependência que não pode ser mantida durante muito mais tempo, sendo que o actual projecto (PEI) será aproveitado neste sentido, para uma documentação dos processos, dos tipos e formas de informação, de requisitos de desenvolvimento, e de tudo o que define o funcionamento deste organismo informático da FCUL.

Ressalta da ideia apresentada até este ponto, que o principal problema no CI é a falta de processos definidos de desenvolvimento, e a forma como todo o crescimento ao nível aplicacional é pouco sustentado por claras metodologias de trabalho. Um dos factores que resulta deste desenvolvimento enfraquecido, é a falta de contextualização de novas plataformas e serviços desenvolvidos e por isso, a FCUL embora tenha neste momento um conjunto significativo de sítios institucionais que servem os seus utilizadores, estes não apresentam qualquer ligação a olhos vistos. A gestão e consulta dos diferentes sítios são realizadas de forma desagregada e sem qualquer ligação explícita nos mesmos. Estes factores fazem com que se preste um serviço isolado e individual por parte de cada um deles, embora semanticamente interligados e pertencentes à mesma instituição.

Sabemos portanto que para além de uma escassez clara em métodos de desenvolvimento bem definidos e de uma predominância de equipas mal estruturadas ou simplesmente mal definidas para o desenvolvimento de serviços, também a gestão de toda a informação existente é bastante dificultada pela desagregação da informação fornecida através dos diferentes sítios de informação e pela falta de estruturação da mesma aquando do desenvolvimento dos próprios serviços. Não existe um ponto comum de desenvolvimento, uma definição clara da estrutura de dados no desenvolvimento, nem tão pouco uma divisão dos vários tipos de tarefas de gestão pelos diferentes tipos de membros da equipa que forma o Centro de Informática. A FCUL não consegue à data de início deste projecto e fazendo uso dos seus meios de informação transmitir correctamente o seu próprio dinamismo para o exterior, tal como não consegue, comunicar de forma ideal no seu interior, havendo demasiados obstáculos no acesso directo e na troca, partilha e gestão de informação interna, bem como na forma como esta é estruturada para divulgação externa através dos seus serviços.

Solução

A proposta de uma nova configuração, com vista à elaboração de uma plataforma comum de desenvolvimento e disponibilização de informações e serviços, vai possibilitar que as restrições que se verificam na actualidade, possam ser devidamente ultrapassadas. Urge que todo o desenvolvimento de diferentes tipos de sistemas de informação seja abordado de uma forma estruturada e baseado em ferramentas que possibilitem às equipas de desenvolvimento do CI, não só um desenvolvimento eficaz, como também um desenvolvimento que proporcione uma facilidade de comunicação interna e um sentido de tecnologia comum a todos os elementos. A gestão de meios de comunicação no CI, deverá passar por um conjunto de regras e de pontos centrais de informação. A partilha e gestão coerente de informação através de diversas fontes de desenvolvimento, permitirá uma reutilização de componentes por diversos sistemas, possibilitando a actualização automática de informação e de recursos replicados por diversos sítios informativos, evitando desagregação e desactualização de informação e contribuindo para um melhor funcionamento dos processos internos à FCUL.

Este projecto deverá focar-se assim primordialmente na constituição de novos métodos de trabalho, métodos esses apoiados em ferramentas suficientemente eficazes e agregadoras de todo o tipo de informação e de todo o tipo de fontes de dados geridos na instituição, resultando na fusão natural entre os variados sítios institucionais actualmente existentes e possibilitando uma evolução para uma futura “porta de entrada única no mundo FCUL”, sendo esta, única, tanto no que diz respeito ao acesso à informação como no que se relaciona com o desenvolvimento de novos serviços ou formas de comunicação.

Resumindo-se a três breves pontos, os principais objectivos deste projecto são:

1. A constituição e/ou adopção de uma plataforma integradora e implementadora de serviços informáticos, adaptável às necessidades dos serviços e dos vários utilizadores existentes.
2. A elaboração de uma metodologia clara de trabalho a implementar no Centro de Informática da FCUL que permita não só comunicação ideal entre toda a equipa de desenvolvimento de

aplicações, como também possibilite uma maior facilidade na transferência de informação entre o núcleo de implementação e os órgãos de gestão da instituição e dos seus diversos órgãos de informação.

3. O mapeamento entre uma metodologia de desenvolvimento de aplicações e a utilização de novas ferramentas que permitam atingir os objectivos de implementação de serviços e aplicações no CI-FCUL.

Destes objectivos e portanto como resultado do respectivo projecto, poderão resultar outros objectivos ou consequências como os seguintes:

1. Validação da utilização abrangente de uma plataforma que possibilite o desenvolvimento futuro de sistemas de informação a serem utilizados na perspectiva de um único contexto FCUL.
2. Elaboração de um processo de substituição de tecnologia e de acessos redundantes aos diferentes meios de informação e comunicação, por agregação num conjunto de mecanismos integrados.
3. Possibilitar a implementação de diferentes serviços de informação baseando-se em apenas uma linguagem de programação, sendo esta utilizada tanto para a codificação dos programas propriamente ditos como também para o acesso aos diversos dados diversificados pelos vários serviços da FCUL.
4. A unificação de uma rede de sítios institucionais que permita facilitar comunicações dentro e entre os diferentes órgãos da FCUL, tendo como ponto de partida o sítio institucional que é raiz da faculdade: www.fc.ul.pt.

1.2.2 Descrição do PEI

Todo o estudo dos métodos de trabalho em execução nesta unidade orgânica de trabalho da FCUL - essencialmente a exaustiva percepção de todas as tarefas necessárias num processo de desenvolvimento interno de aplicações -, bem como a análise das formas de comunicação adoptadas entre os diversos elementos que formam o CI e das ferramentas de trabalho utilizadas no desenvolvimento de serviços, representam todo um estudo primordial integrante deste Projecto de Engenharia Informática:

As funções a desempenhar pelo integrante focar-se-ão assim, em primeira mão, na análise dos sistemas de informação e ferramentas de trabalho existentes nos vários serviços, seguindo-se um estudo intensivo de várias metodologias de trabalho utilizadas por diversas organizações e que permitem sustentar um modelo de desenvolvimento bem definido e documentado. Tarefa procedente a estes envolventes estudos é a de definição e documentação de toda uma metodologia de trabalho eficaz (e respectivas ferramentas e documentos) no contexto de desenvolvimento em que o Centro de Informática se encontra, bem como proceder ao levantamento das condições ou obstáculos necessários de criar ou ultrapassar (respectivamente) para que toda a metodologia definida seja adoptada na prática pela instituição.

Após todo o processo de análise ser concluído e na fase final do estudo das diferentes ferramentas de desenvolvimento a adoptar, o candidato teve ainda oportunidade de realizar algumas tarefas nas fases

de implementação de um projecto experimental, tendo estado no entanto esta oportunidade, dependente das datas referentes è entrega do Projecto de Engenharia Informática. Estas tarefas de implementação prenderam-se essencialmente com a realização de pequenas funcionalidades, ou conjuntos de funcionalidades, que têm como principal objectivo demonstrar a utilização de novas ferramentas adoptadas e provar que a integração entre estas é válida para um desenvolvimento futuro de acordo com os objectivos de todo este projecto.

Nota: A necessidade de se proceder a um levantamento exaustivo das verdadeiras necessidades de divulgação de informação, utilizando diversos meios disponíveis, entre os quais a avaliação de estatísticas de acesso para averiguar o reaproveitamento de sítios ou a sua completa reestruturação, representava também a início, um ponto base para que o projecto pudesse ser realizado, no entanto com o avançar do projecto, este ponto foi excluído das tarefas do projecto, em detrimento de uma maior aprofundamento nos casos de estudo das diversas ferramentas utilizadas para desenvolvimento.

1.2.3 Discussão do plano de trabalhos e objectivos:

É importante referir que neste capítulo de caracterização e calendarização de tarefas realizadas durante a extensão deste projecto, faz-se uma separação entre dois planos de trabalhos ligeiramente diferentes, o primeiro refere-se ao planeamento realizado aquando do início do projecto, enquanto o segundo indica o plano de trabalhos resultante de uma alteração de prioridades e de tempos de esforço, alteração esta efectuada e acordada entre as várias partes envolvidas já no decorrer do projecto.

Caracterização e Calendarização de Tarefas a Realizar

Plano de Trabalhos Inicial – Setembro de 2009

Tarefa	Tempo Esperado (em meses)	Desvio Máximo (em meses)
1. Análise dos diferentes processos funcionais e necessidades residentes nas unidades orgânicas FCUL.	0	0
2. Estudo de diferentes metodologias de desenvolvimento existentes e suas possibilidades de adopção no Centro de Informática	2	0.5
3. Análise de soluções tecnológicas open-source que visarão facilitar a implementação de novos serviços de informação, assim como, a partilha de informação e a adaptação dos serviços existentes.	1.5	0.5
4. Análise social centrada nos utilizadores dos diferentes serviços, para percepção de padrões de utilização e de formas de desenho mais apropriadas, bem como, de falhas existentes na sua utilização actual.	0.5	0.25
5. Análise de vantagens e desvantagens da execução do projecto.	1	0.50
6. Implantação e integração das soluções tecnológicas de forma a disponibilizar o futuro desenvolvimento de serviços e sítios institucionais.	4	2

Total de Tempo de Realização do Projecto:	**9**	**3.75**

Tabela 1: Plano de trabalhos inicial – Setembro 2009

Plano de Trabalhos Revisto – Abril de 2010

Tarefa	Tempo Verificado (em meses)	Tempo Esperado (em meses)
1. Análise dos diferentes processos funcionais e necessidades residentes nas unidades orgânicas FCUL.	0	0
2. Estudo de diferentes metodologias de desenvolvimento existentes e suas possibilidades de adopção no Centro de Informática	2.5	2
3. Análise de soluções tecnológicas open-source que visarão facilitar a implementação de novos serviços de informação, assim como, a partilha de informação e a adaptação dos serviços existentes.	2.0	1.5
7. Definição da metodologia de desenvolvimento a implementar no Centro de Informática (com construção de modelos de documentos tipicamente gerados por esta).	0.5	--
5. Análise de vantagens e desvantagens da execução do projecto.	1	1
6. Realização e Implementação de casos de teste que visam a implantação e integração das soluções tecnológicas de forma a disponibilizar o futuro desenvolvimento de serviços e sítios institucionais.	3	4
Total de Tempo de Realização do Projecto:	**9**	**8.5**

Tabela 2: Plano de trabalho revisto – Abril 2010

Discussão de Tarefas dos Planos de Trabalho

Tarefa 1. Igual em ambos os planos, prende-se com toda a análise necessária de realizar pelas diversas unidades integrantes da Faculdade de Ciências da Universidade de Lisboa, faculdade em que o Centro de Informática está inserido e para a qual desempenha as suas funções. Será difícil traduzir esta fase num estudo preciso, no entanto e uma vez que estando eu inserido na própria faculdade também como aluno, existe já à data de começo do projecto, consciência de todos os processos funcionais da FCUL que giram em torno dos seus estudantes, órgãos, departamentos, funcionários, etc., tal como existe consciência do conjunto de sistemas de informação e serviços Web oferecidos pelo CI. Assim, e embora lhe seja atribuída uma duração de 1 mês (duração simbólica), este período já foi cumprido através do estágio efectuado previamente na mesma instituição (estágio este que já foi referido no enquadramento do PEI).

Tarefa 2. Definida em ambos os planos de tarefas acima especificados, esta tarefa dá-se por concluída com uma definição exaustiva e precisa de uma metodologia ágil de desenvolvimento e

das várias fases que a constituem. Metodologia esta que será o resultado de uma combinação de várias metodologias estudadas, uma vez que entendemos que nenhuma metodologia seria por inteiro ideal para implementação no Centro de Informática. No entanto, informação mais aprofundada sobre cada metodologia estudada, e sobre os métodos adoptados serão explicados mais à frente neste relatório. De referir ainda, que esta tarefa sofreu na prática um incremento de meio mês no que era originalmente a sua duração prevista, estendendo-se durante os iniciais 2.5 meses de projecto e ainda na definição de uma outra tarefa (tarefa 7).

Tarefa 3. Em terceiro lugar, e coincidente também em ambas as planificações, temos a realização de um estudo de formas de implementação simplificada de novos serviços (bem como integração dos já existentes) no CI-FCUL, através da adopção e utilização de ferramentas open-source. Este representa um dos passos mais importantes em todo o projecto, uma vez que a escolha errada da tecnologia a usar na implementação futura de serviços é o primeiro grande passo para um rotundo fracasso. Neste estudo, terá que ser sempre levado em conta, o facto de ambas as tecnologias escolhidas terem de ser necessariamente capazes de serem integradas para um desenvolvimento conjunto, para que qualquer serviço ou aplicação desenvolvida não esteja representado sobre a forma de uma linguagem que mais tarde se torna incomportável de reutilizar. Sabe-se também, que para o Centro de Informática interessam dois tipos de ferramentas de desenvolvimento, um Gestor de Conteúdos (CMS) capaz de gerir informação de uma forma simplificada e desenvolver conteúdos simples de forma intuitiva (e para não programadores), e uma ferramenta de desenvolvimento de conteúdos mais complexos capaz de ser devidamente trabalhada pelos programadores e de integrar conteúdos desenvolvidos no CMS. Finalmente, interessa referir que para realização desta tarefa é necessário, algum envolvimento dos vários elementos constituintes do Centro de Informática, no sentido de aprovação e de escolha das ferramentas que mais convêm a todos.

Tarefa 4. O principal ponto de ruptura no plano inicial de trabalhos encontra-se neste ponto. Previsto inicialmente como uma "Análise social centrada nos utilizadores dos diferentes serviços, para percepção de padrões de utilização e de formas de desenho mais apropriadas, bem como, de falhas existentes na sua utilização actual" mas abandonada por compreensão de uma maior necessidade na " Definição da metodologia de desenvolvimento a implementar no Centro de Informática (com construção de modelos de documentos tipicamente gerados por esta)." (tarefa 7). Isto deveu-se ao facto do estudo inicial das metodologias de desenvolvimento ter sido finalizado com a definição de pontos que possibilitavam a percepção de quais as metodologias que mais hipóteses apresentavam de ser adoptadas no Centro de Informática, mas sem que a definição da própria metodologia e de todos os pormenores relacionados com esta (como o processo de desenvolvimento e os modelos de documentos gerados) fosse de facto definido. Outra razão que levou a este ajuste no planeamento, relaciona-se com o facto de todo o trabalho relacionado com a integração das ferramentas e com a realização de casos de teste de implementação de serviços (tarefas seguintes a esta), terem obrigatoriamente de ser realizadas para que o projecto tenha sucesso, forçando a que o tempo de realização restante no projecto,

não fosse suficiente para que a tarefa original de pesquisa de padrões de utilização se traduzisse em realidade.

No entanto, interessa referir que esta tarefa se relacionava com a compreensão do funcionamento de cada departamento pertencente à FCUL e formas directas ou indirectas como são afectados pelos diversos mecanismos de informação existentes na instituição. Para além disso, compreendia ainda a realização de um estudo sobre projectos anteriores focando-se na forma como nestes se partiu para a implementação, bem como a realização de pequenas entrevistas junto dos principais responsáveis e principais utilizadores dos diferentes serviços. Teria assim como objectivo entender a forma geral como todos os departamentos se relacionam entre si e com a faculdade através dos sistemas de informação existentes.

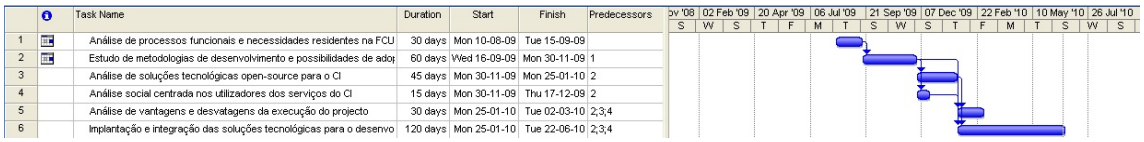
Tarefa 5. Presente no planeamento inicial e mantido no plano de trabalhos revisto, a "Análise de vantagens e desvantagens da execução do projecto" é naturalmente essencial para a justificação de todo este projecto. Representa um remate para todo o projecto e tem como objectivo único, enumerar todos os ganhos e perdas imediatos e projectar todos os ganhos e perdas futuros, resultantes da implementação de uma nova metodologia de trabalhos e do uso de novas ferramentas de desenvolvimento, pelo Centro de Informática, bem como perceber quais as principais barreiras a ultrapassar no futuro como garante para que todo o trabalho realizado no projecto não tenha sido em vão. Naturalmente esta tarefa estende-se para lá deste projecto, e procura perceber o futuro, com base no que é sugerido e perceptível após término de todas as tarefas. Por isso, relaciona-se naturalmente com as tarefas que lhe seguem, de implementação e integração das ferramentas adoptadas e dos casos de estudo de implementação, uma vez que só durante a realização destes se vai ganhado consciência do real valor de todo o projecto.

Tarefa 6. Sem grandes alterações na revisão do plano de trabalhos, a sexta tarefa deste projecto relaciona-se com a implantação e integração das diferentes ferramentas a adoptar, bem como com a realização de casos de teste que permitem provar as mesmas como certas para o desenvolvimento futuro no Centro de Informática. A redução desta fase em um mês de duração prende-se com o facto das tarefas iniciais do projecto se terem estendido durante mais tempo do que o inicialmente previsto, no entanto como poderá ser provado mais à frente neste documento, a redução do tempo desta tarefa não prejudicou de forma substancial os objectivos da mesma.

Tarefa 7. Como já foi referido na descrição da tarefa 4, a agregação específica de tempo a uma nova tarefa de definição da metodologia a adoptar no Centro de Informática prende-se com o facto do estudo metodológico se ter estendido mais do que era de prever a início, mas também se prende com o facto de uma maior profundidade ao nível desta definição permitir uma melhor percepção da mesma e propiciar a sua adopção no CI. A relação directa entre a definição de modelos documentais de um típico desenvolvimento de projecto, e as tarefas que se seguem relacionadas com a implementação de casos de estudo usando as diferentes ferramentas de desenvolvimento permite testar de forma mais fiel toda a metodologia definida e ter uma ideia mais fiel das vantagens e desvantagens da metodologia e ferramentas adoptadas. Devo referir

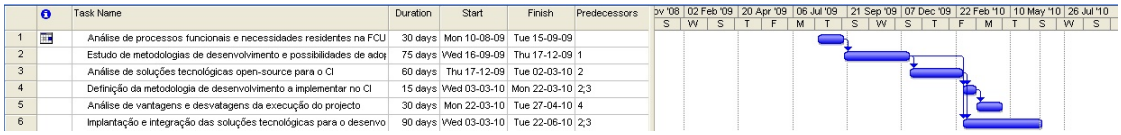
por fim que o tempo de realização para a definição da metodologia de desenvolvimento é o mesmo que o da tarefa original (tarefa 4), estendendo-se por meio mês.

Uma vez que a realização das tarefas acima enumeradas não é sempre levada a cabo de forma sequencial, sendo muitas vezes efectuadas de forma paralela, deve-se entender que a sua duração real individual poder-se-á traduzir em números ligeiramente superiores ao estimado em termos de datas de conclusão (referidas na proposta de projecto), sendo que a sua duração se mantém no entanto praticamente inalterada. Para que este facto seja mais transparente, de seguida comparo ao pormenor os dois planos em termos de datas de início e fim das suas tarefas e respectiva duração, fazendo uso de mapas de Gantt (Figura 1e Figura 2):



	Task Name	Duration	Start	Finish	Predecessors
1	Análise de processos funcionais e necessidades residentes na FCU	30 days	Mon 10-08-09	Tue 15-09-09	
2	Estudo de metodologias de desenvolvimento e possibilidades de adop	60 days	Wed 16-09-09	Mon 30-11-09	1
3	Análise de soluções tecnológicas open-source para o CI	45 days	Mon 30-11-09	Mon 25-01-10	2
4	Análise social centrada nos utilizadores dos serviços do CI	15 days	Mon 30-11-09	Thu 17-12-09	2
5	Análise de vantagens e desvatagens da execução do projecto	30 days	Mon 25-01-10	Tue 02-03-10	2;3;4
6	Implantação e integração das soluções tecnológicas para o desenvo	120 days	Mon 25-01-10	Tue 22-06-10	2;3;4

Figura 1: Tarefas e mapa de Gantt respectivo do Plano de Trabalhos Inicial



	Task Name	Duration	Start	Finish	Predecessors
1	Análise de processos funcionais e necessidades residentes na FCU	30 days	Mon 10-08-09	Tue 15-09-09	
2	Estudo de metodologias de desenvolvimento e possibilidades de adop	75 days	Wed 16-09-09	Thu 17-12-09	1
3	Análise de soluções tecnológicas open-source para o CI	60 days	Thu 17-12-09	Tue 02-03-10	2
4	Definição da metodologia de desenvolvimento a implementar no CI	15 days	Wed 03-03-10	Mon 22-03-10	2;3
5	Análise de vantagens e desvatagens da execução do projecto	30 days	Mon 22-03-10	Tue 27-04-10	4
6	Implantação e integração das soluções tecnológicas para o desenvo	90 days	Wed 03-03-10	Tue 22-06-10	2;3

Figura 2: Tarefas e mapa de Gantt respectivo do Plano de Trabalhos Revisto

Como se pode ver pelas duas figuras acima, as principais diferenças nos planos de trabalho especificados começam aquando do início e fim da terceira tarefa ("Análise de Soluções Tecnológicas open-source para o CI"). No primeiro plano esperava-se o início de uma análise centrada nos utilizadores, mas no entanto, o que acabou por se verificar foi o cancelamento dessa mesma análise por extensão da terceira tarefa e necessidade de uma especificação maior da metodologia de desenvolvimento a adoptar. Pode-se verificar ainda que a extensão da terceira tarefa em termos de duração provocou que a duração da sexta tarefa fosse reduzida substancialmente para que os prazos do projecto não fossem ultrapassados. Em ambos os planos a tarefa de análise de vantagens e desvantagens processa-se com o fim da definição da metodologia a adoptar e com o início da realização dos casos de estudo sobre as ferramentas de desenvolvimento. Para além destas indicações, a leitura directa dos dois gráficos, ajuda a perceber os pormenores relacionados com datas de começo e fim das diferentes tarefas e com a evolução que o projecto sofreu nos diferentes meses de realização do PEI.

1.3 Estrutura da Tese

Este documento está organizado da seguinte forma:

Capítulo 2: Análise de Metodologias de Desenvolvimento – Neste segundo capítulo é feito um estudo elaborado sobre metodologias de desenvolvimento, em que são abordados oito métodos bem conhecidos no mundo empresarial dos dias de hoje. São identificadas as características de cada metodologia, bem como o processo que guia a implementação de novas aplicações, os papéis a desempenhar pelos membros das equipas de desenvolvimento, e as práticas comuns, as experiências de utilização e os pormenores de adopção de cada metodologia. No final de cada análise a uma metodologia, são indicados os factores a favor e os factores contra a sua adopção no CI-FCUL.

Capítulo 3: Definição de Uma Metodologia de Desenvolvimento – A definição de uma metodologia de desenvolvimento para adopção no Centro de Informática, com o objectivo de tornar o desenvolvimento de projectos um processo mais eficaz e mais organizado é especificada ao pormenor neste capítulo. Assim sendo, o capítulo apresenta o processo de desenvolvimento que caracteriza a metodologia resultante do estudo presente no capítulo anterior, bem como define os papéis a desempenhar pelos diferentes membros da equipa de desenvolvimento e ainda especifica modelos de documentos a serem gerados na elaboração de cada projecto segundo a metodologia.

Capítulo 4: Análise de Ferramentas – Após o estudo de metodologias e a definição da metodologia a adoptar no CI-FCUL, segue-se a análise das ferramentas a utilizar no desenvolvimento de projectos e portanto a integrar na metodologia de trabalhos adoptada. Este estudo estende-se por dois tipos de ferramentas: uma primeira denominada como plataforma de gestão de conteúdos Web que tem por objectivo a construção e gestão de conteúdos Web de natureza simples (conteúdos estáticos) facilitando a edição e acesso a conteúdos aos utilizadores com menos conhecimentos programáticos bem como a programadores, e possibilitanto a integração destes mesmos conteúdos com conteúdos mais complexos; e uma segunda denominada como Plataforma Rápida de Desenvolvimento que tem por objectivo a implementação de aplicações, sítios ou serviços Web de natureza mais complexa e que possibilitam o uso de dados presentes em fontes externas (como bases de dados ou exportação de dados de outras fontes) criando conteúdos dinâmicos e inteiros sistemas de serviço-resposta na Web. O estudo de um gestor de conteúdos adequado ao CI é feito através da revisão e análise de uma dezena de plataformas e sua avaliação no que diz respeito a um conjunto de características essenciais ao Centro, entre as quais estão, a facilidade de edição de conteudo, o tipo de acesso e partilha de conteúdos disponibilizado pela plataforma, pormenores relacionados com a segurança, pormenores relacionados com a capacidade de integração da plataforma com outras ferramentas, etc. No que diz respeito à plataforma de desenvolvimento é especificada a plataforma adoptada desde o início do projecto, as razões de adopção desta plataforma, as suas principais características e modos de funcionamento, sendo ainda avançadas outras plataformas que poderiam ter sido adoptadas como alternativa.

Capítulo 5: Uso de Ferramentas para Desenvolvimento Futuro de Aplicações - Após especificação e escolha de ferramentas a usar no desenvolvimento, interessa provar que a sua utilização permite preencher todos os requisitos de desenvolvimento e de processos, e que ambas as tecnologias são possíveis de integrar num desenvolvimento centrado e com pontos de acesso comuns. A realização de

casos de estudo fazendo uso das diversas ferramentas é por isso essencial para a percepção de todo o funcionamento e formas de desenvolvimento futuro de aplicações.

Capítulo 6: Mapeamento das Ferramentas de Trabalho na Metodologia Adoptada – Estando definida a metodologia a adoptar no desenvolvimento de aplicações e serviços no CI-FCUL, bem como estando descobertas as ferramentas a utilizar, neste capítulo é especificado o mapeamento das últimas na primeira, e portanto, de que forma se faz a ligação entre métodos e ferramentas de implementação.

Capítulo 7: Conclusão – Neste capítulo que fecha este relatório de Projecto de Engenharia Informática, são indicadas e discutidas todas as conclusões a que foi possível de chegar durante a realização do projecto. Tanto no que diz respeito às metodologias abordadas, como às ferramentas estudadas, como a factores relacionados com o futuro do CI-FCUL, todos os pormenores importantes são identificados, sendo ainda avançadas possíveis tarefas futuras sobre o projecto e a forma como contrubuíriam para este.

Capítulo 2 Análise de Metodologias de Desenvolvimento

2.1 Introdução

Os métodos de desenvolvimento de software procuram constituir uma resposta para todo o mundo de negócios reduzindo a carga de trabalhos que se verifica ao nível da implementação de serviços e aplicações, com a definição de processos de desenvolvimento suficientemente rápidos e ágeis para suportarem mudanças num mundo tecnológico em constante mudança.

Os métodos tradicionais de trabalho e a forma como se baseiam muitas vezes em ideais inatingíveis que procuram ligar processos de desenvolvimento a situações funcionais utópicas, são capazes de levar inteiras empresas a situações insustentáveis e a um crescimento apoiado em falsas expectativas. Esta situação, verificada a nível global, em diversas organizações, levou ao surgimento de metodologias ágeis de desenvolvimento de software, um tipo de metodologia que nasce da ideia de que não existe um processo capaz de servir qualquer tipo de projecto, e que por isso as práticas devem ser adoptadas de forma a servirem as necessidades de cada projecto.

Não é aliás linear, a diferença entre os métodos introduzidos pelo desenvolvimento ágil de software e as aproximações mais tradicionais - os limites nunca foram claramente estabelecidos -, mas é possível fazer distinções entre os dois tipos de métodos partindo da análise do tradicional modelo cascata para desenvolvimento aplicacional, e apontando quais os seus principais pontos fracos aos quais os novos métodos procuram dar uma resposta inteligente.

2.1.1 Método Cascata como método utópico e Métodos ágeis como uma resposta à realidade:

O método denominado como “Cascata” - ou em Inglês “waterfall” – é um método em série para desenvolvimento de projectos que se baseia em fases bem definidas que se seguem sequencialmente umas às outras, sendo que cada uma apenas pode avançar quando a anterior está já inteiramente concluída. A ordem e as fases processam-se da seguinte forma (Figura 3):

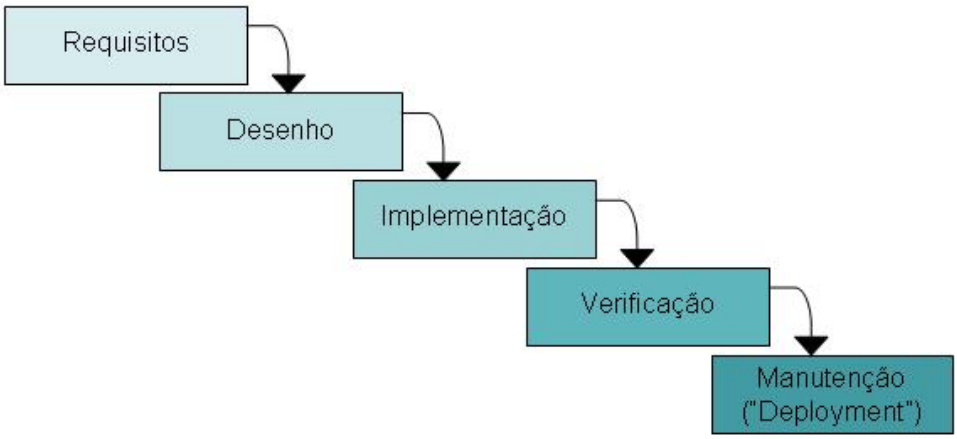


Figura 3: ***Esquema exemplificativo do método "Waterfall"***

Podemos dizer que de uma forma geral, e até ao aparecimento de metodologias ágeis, esta era a sequência de tarefas seguida pela maioria das empresas, tendo-se verificado uma taça de sucesso inferior aos 50***%***. Pensou-se primeiro que o insucesso pudesse ser evitado se as duas primeiras fases (requisitos e desenho) fossem aprofundadas na sua elaboração para tentar construir uma base de implementação com menores margens de erro, no entanto, e após alguns anos de intensivas especificações de requisitos, verificou-se que erros feitos nestas primeiras etapas de desenho apenas eram descobertos já durante a implementação ou lançamento (do inglês "Deployment") de todo o projecto, altura essa, em que se tornavam demasiado caras quaisquer tentativas de recuperação. ***[Serena, 2007]***

O método Cascata parte do pressuposto de que é possível existir um perfeito entendimento dos requisitos desde o início do projecto, quando na realidade o que se verifica na maior parte dos casos, é o cliente ou o responsável pelo projecto (ou ambos) não saberem bem o que desejam ver implementado, levando a uma incapacidade de articular os requisitos com vista ao sucesso do desenvolvimento. ***[Kramtchenko, 2004]***

O facto de no método tradicional as diferentes fases estarem como que separadas por etapas de conclusão, e de nos métodos ágeis não chegarem bem a existir fases de desenvolvimento, mas sim iterações que procuram produzir código funcional em resposta a cada funcionalidade que se procura implementar, ou em resposta a cada mudança verificada ao nível dos requisitos, marca a principal diferença entre os dois diferentes tipos de metodologias de desenvolvimento: "*Existe uma grande distinção entre desenvolvimento planeado e desenvolvimento ágil…No primeiro tendemos para processos controlados, sequenciais, replicáveis e orientados a objectivos; enquanto no segundo consideramos principalmente processos orientados por uma aleatoriedade ou um oportunismo acidental e que são na sua maior parte simultâneos ou separados por curtos intervalos de tempo e sempre produto de uma negociação ou compromisso caprichoso*". ***[McCauley, 2001].***

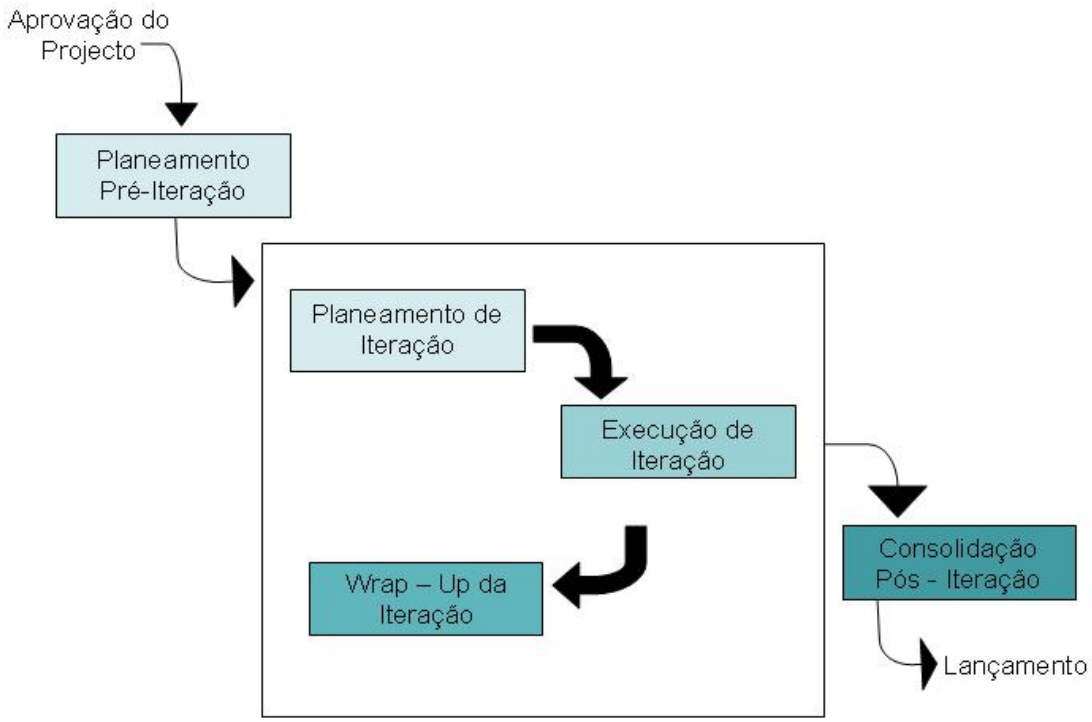


Figura 4: ***Esquema exemplificativo do processamento de uma iteração num método ágil.***

No desenvolvimento ágil (Figura 4), flexível e adaptável é objectivo que cada iteração demore relativamente pouco tempo (1 a 3 semanas) e inclua todos os passos do método cascata tradicional, produzindo rapidamente e como resultado, formas visuais de representação e verificação de cada funcionalidade implementada. Desta forma, tanto os erros cometidos em fases de desenho passam a ser descobertos em etapas prematuras de desenvolvimento, como também é possível para os intervenientes directamente envolvidos na definição de requisitos redefinirem estes, ainda em tempo útil de desenvolvimento, de acordo com os vários factores em mudança ***[Williams, 2007; Serena, 2007]***.

Concluindo numa perspectiva de visão do cliente num projecto que use métodos ágeis de desenvolvimento, este pode especificar os seus requisitos para a próxima versão do produto baseando-se na observação da evolução de um produto em vez de especular sobre o que é necessário aquando do início do projecto.

2.1.2 Surgimento da "Agilidade":

O surgimento da "agilidade" em metodologias de desenvolvimento informático foi proporcionado, como já referi, por razões que se prendem com a impraticabilidade dos métodos clássicos de desenvolvimento face às mudanças constantes de requisitos em ambientes em constante alteração, principalmente no que diz respeito a aplicações e a software baseado em Internet. Possibilitou esta evolução, o aparecimento de métodos mais ágeis responsáveis por iterações curtas e de menor

complexidade e risco, melhor feedback e maior produtividade acompanhada por uma taxa de sucesso mais elevada na construção deste tipo de sistemas aplicacionais.

O "Manifesto para Desenvolvimento Ágil de Software" desencadeado (por volta de 2001) por um grande número de organizações e investigadores - denominada Aliança Ágil - apoiou-se em valores bem definidos que se traduzem em diversas regras de desenvolvimento ***[Beck et al, 2001; Cockburn, 2002]***:

1. **Indivíduos e interacções** VS processos e ferramentas
    a. Promover relações próximas entre vários elementos de equipa de desenvolvimento.
    b. Possibilitar um ambiente de trabalho fechado e coeso.
    c. Construir projectos à volta de individualidades motivadas, fornecendo-lhes o ambiente, o suporte e a confiança necessários para que o trabalho seja desenvolvido.
    d. Limitar ao mínimo a institucionalização e seus procedimentos.
    e. Reflexão por parte da equipa sobre como se tornar mais eficaz deve ser realizada de tempos a tempos.
2. **Ferramentas funcionais** VS Documentação compreensiva
    a. Fazer contínuos testes funcionais de software
    b. Produzir novas versões da aplicação frequentemente
    c. Programadores devem escrever código simples, e tecnicamente avançado, definindo regras de programação claras para toda a equipa.
    d. Reduzir a documentação apenas ao essencial.
    e. Dedicar uma atenção contínua às boas técnicas e a bons desenhos promove a agilidade aplicacional e de requisitos.
3. **Colaboração dos Clientes** VS Negociação de contratos
    a. Relacionamento e cooperação entre desenvolvedores e clientes.
    b. Processo de negociação visto como uma forma de atingir uma relação viável.
    c. Orientação para apresentação de resultados imediatamente após o início do projecto.
    d. Centrar a prioridade na satisfação do cliente.
4. **Resposta a mudanças** VS Seguir um plano
    a. Partes envolvidas devem estar sempre bem informadas, competentes e preparadas para implementar possíveis ajustes durante o processo de desenvolvimento do projecto.
    b. Celebração de contratos deve ser feita de forma a possibilitar mudanças no processo de desenvolvimento sempre que destas mudanças advêm claras vantagens para os clientes.

Todos os valores evidenciados no lado esquerdo de cada ponto da listagem acima especificada, indicam prioridades a ser seguidas na metodologia de desenvolvimento ágil de software, implementado ao nível de uma organização, em detrimento de todos os valores que com estes se confrontam e que se situam do lado direito de cada ponto.

A essência de um método de desenvolvimento ágil de software resume-se ao uso de um número leve, mas suficiente, de regras de comportamento do produto orientadas aos seres humanos e à comunicação entre estes.

2.1.3 Caracterização de Metodologia Ágil de Desenvolvimento:

De tudo o que foi referido em cima é relativamente fácil ficar com as seguintes ideias:

O desenvolvimento ágil de software é essencialmente incremental, cooperativo, simples e adaptativo. Os seus métodos são portanto principalmente simples e rápidos e o grupo de desenvolvimento caracteriza-se por se concentrar, em primeiro lugar, nas funções tidas como necessárias, conduzindo a um rápido produto/protótipo, recolhendo feedback e reagindo à informação recebida para construção de versões futuras.

Retém-se também que os requisitos são sempre introduzidos, modificados e removidos em iterações sucessivas, e que portanto, o ambiente em constante mudança leva a alterações nas especificações iniciais adiantadas para um projecto. Satisfazer o cliente na altura de lançamento das versões da aplicação é mais importante que o satisfazer aquando do início da mesma e há que centrar esforços em não colocar entraves a alterações de especificações, arranjando forma de responder adequadamente a essas alterações tendo em vista o objectivo do cliente. O mundo ágil é assim composto por um conjunto de pessoas que estão orientadas para a eficácia e para a mudança ou adaptação e que lidando com essa receptibilidade à mudança se tornam capazes de levar o projecto a bom porto.

A simplicidade, ou arte de maximizar a quantidade de trabalho feito, é essencial em todo um processo deste tipo. Equipas organizadas são as únicas capazes de retirar o máximo proveito das melhores arquitecturas, requisitos e desenhos e a orientação às pessoas permite reduzir o tempo de aprendizagem, os riscos e a incapacidade produtiva.

Assim, em toda a sua essência, uma metodologia ágil de desenvolvimento, é todo um processo incremental e convergente, construído em pequenos passos e capaz de minimizar dessa forma os riscos de desenvolvimento, tornando-se um processo mais flexível e por isso mais adaptativo. Procura-se sempre uma vitória prematura, através de soluções simples e de um aumento da qualidade de desenho de forma contínua (iteração para iteração) acompanhando o desenvolvimento de contínuos testes que impedem a irreversibilidade trágica do projecto.

Resumindo toda esta caracterização a um conjunto de pontos simplificados, sabe-se linearmente que uma Metodologia Ágil de Desenvolvimento, tendo como ponto de vista a obtenção de rápidos resultados, apresenta [***Miller, 2001; Ambler, 2002; Highsmith & Cockburn, 2001]***:

1. Modularidade no processo de desenvolvimento.
2. Processo Iterativo com pequenos ciclos, rápidas verificações e correcções.
3. Ligação a prazos de tempo, definindo ciclos de iteração com 2 a 8 semanas.
4. Parcimónia no processo de desenvolvimento, evitando actividades desnecessárias e optando pela simplicidade.
5. Desenvolvimento adaptativo com mitigação de possíveis riscos emergentes.
6. Processo incremental e convergente (através de prototipagem).
7. Trabalho tipicamente colaborativo e comunicativo entre elementos de equipa.
8. Orientação às pessoas e aos clientes.
9. Menor documentação possível.

Formas de aumentar a qualidade em desenvolvimento ágil:

Existem sempre factores a ter em conta no tipo de metodologia de desenvolvimento que acabou de ser descrita. Este processo simultaneamente ágil e capaz de satisfazer os objectivos que lhe são propostos, pode ser acelerado através de um conjunto de "Sweet Spots" (requisitos) de desenvolvimento ágil definidos por Cockburn em 2002**.** Ter em conta que a presença de cada um dos factores que são a seguir referidos melhora as hipóteses de sucesso da metodologia a ser adoptada ***[Cockburn, 2002]***:

- Duas (2) a oito (8) pessoas em uma sala promovem a comunicação e o trabalho contínuo em equipa.
- Uso de especialistas no local de trabalho (em inglês "On-site usage experts") permite acelerar funcionalidades com que a equipa está menos familiarizada.
- Incrementos curtos com pequenos ciclos de testes, feedback e reparação, são essenciais a um desenvolvimento sem erros.
- Desenvolvedores experientes permitem diminuir o tempo de desenvolvimento.
- Uso de ferramentas de simplificação de tarefas deve ser procurado como forma de aliviar as despesas de programação.
- Uso de standards de código permite homogeneizar o desenvolvimento.

A qualidade do software produzido por metodologias ágeis de desenvolvimento é ainda possível de aumentar através do uso de técnicas tipicamente presentes em vários métodos aplicativos desta metodologia, como a prática de "Refactoring" ou de "Pair Programming" presentes no método de Programação Extrema. No entanto, uma exposição mais alongada de todas estas técnicas, será feita aquando do debate e da exploração científica individual de cada método, mais à frente neste projecto.

2.1.4 Princípios de Metodologia Ágil: Programação Pragmática

A ter sempre em consideração em qualquer metodologia de desenvolvimento adoptada, estão também os princípios de programação pragmática, um conjunto de práticas de programação publicadas por Andrew Hunt e David Thomas em "The Pragmatic Programmer" ***[Hunt, 2000]*** que passaram a ser também elas incluídas na filosofia da metodologia ágil de desenvolvimento. Cobrindo práticas de programação de uma forma bastante extensa e de uma maneira bem fundada, define seis indispensáveis dicas de programação:

- Membros da equipa devem assumir sempre as responsabilidades pelos seus actos, tal como devem pensar sempre em soluções para resolver eventuais problemas em vez de se resguardarem em desculpas.
- Não deve nunca ser dado suporte a programação e desenho mau ou inconsistente, devendo-se corrigir essas irregularidades sempre que são encontradas.
- Sempre que for necessário, qualquer membro da equipa deve participar de forma activa na introdução da mudança ao nível do desenvolvimento. Como exemplo, se uma ferramenta de desenvolvimento deixa de ser suficiente para suportar o desenvolvimento, dever-se-á procurar uma nova solução e cada membro da equipa deverá ir ao encontro desta e não ficar "preso" às ferramentas com que está familiarizado
- Produzir software que agrade ao cliente, mas saber quando parar. Não esquecer que o conhecimento tecnológico está do lado da equipa de desenvolvimento e procurar satisfazer as exigências do cliente até a satisfação dos requisitos da aplicação estar garantida, não favorecendo extravagâncias.
- Transmitir sabedoria constantemente. Tanto entre os vários elementos da equipa de desenvolvimento, como com o próprio cliente. Não cometer o erro de querer o conhecimento apenas para o próprio, e procurar a discussão de ideias com restantes elementos sempre que existirem dúvidas no âmbito do projecto.
- Melhorar formas de comunicação. Procurar implementar um ambiente e sistema de comunicação, que possibilite uma forma constante de trabalho ao mesmo tempo que possibilita a natural comunicação entre os vários elementos da equipa de desenvolvimento, tal como com os restantes intervenientes indirectos.

2.2 Características Importantes para Implementação de uma Metodologia no CI

Uma temática importante de abordar antes de passar ao estudo de várias metodologias de desenvolvimento ágil, é a de levantamento do conjunto de características importantes de observar numa metodologia para que possa ser adoptada no Centro de Informática da FCUL. Assim sendo, é importante que a metodologia de desenvolvimento implemente as seguintes características:

- Processo de desenvolvimento baseado em curtos ciclos iterativos de desenvolvimento de funcionalidades, caracterizado pelo desenvolvimento de um protótipo que vai sendo constantemente enriquecido com a implementação das diversas funcionalidades.

- Equipa de desenvolvimento relativamente curta, entre 2 e 10 pessoas, e em que todos partilham o mesmo espaço físico, ou encontram-se bastante próximos entre si.

- Processo de desenvolvimento orientado ao Cliente, possibilitando que os objectivos da aplicação, bem como a ordem pelas quais as funcionalidades são desenvolvidas, satisfaçam este.

- Processo de recolha de requisitos muito completo por forma a que todo o desenvolvimento seja guiado na direcção certa desde o início, e apoiado no facto de a maior parte dos clientes das aplicações e serviços desenvolvidos pelo CI-FCUL, não terem um papel muito activo no desenvolvimento, entrando em contacto com a equipa de desenvolvimento tipicamente em alturas de começo, meio e fim do projecto.

- Processo de desenvolvimento orientado para uma boa documentação de cada projecto desenvolvido, tanto em termos de planeamento, como em termos de progresso e testes a funcionalidades (sem que no entanto essa documentação se torne excessiva).

- Existência de mecanismos mínimos de controlo de tempo e custos do projecto, através essencialmente de reuniões ou revisões de versões ou funcionalidades por parte da equipa de desenvolvimento. Existência ainda, de mecanismos de monitorização de todo o progresso no desenvolvimento dos serviços ou aplicações.

- Divisão de papéis e responsabilidades pelos membros da equipa de desenvolvimento feita de forma relativamente simples, em que existe tipicamente um membro com maiores responsabilidades no projecto, e com maior poder de decisão, e em que os restantes reportam o seu trabalho a este, recebendo dele instruções sobre as tarefas a realizar.

- Semanas de trabalho com cerca de 40 horas, equivalendo a 8 horas por dia por trabalhador.

Na análise que se segue referente a vários tipos de metodologias e suas hipóteses de adopção ou de integração numa metodologia resultante para adopção no CI-FCUL, é considerado que a existência dos pontos acima em cada metodologia, representará obviamente uma vantagem na sua adopção, uma vez que contribui para a metodologia ideal pretendida. A existência de todos os pontos acima referidos numa das metodologias que a seguir se descrevem, torna-a naturalmente ideal para adopção no CI-FCUL.

2.3 Metodologias Ágeis de Desenvolvimento Abordadas:

A adopção de uma metodologia ágil de desenvolvimento é necessáriamente precedida de um estudo comparativo elaborado, que procura analisar várias metodologias e perceber quais aquelas cujas características, práticas e regras, são de facto capazes de responder às necessidades existentes na organização. Neste capítulo procuro documentar esse mesmo estudo para o Centro de Informática da Faculdade de Ciências da Universidade de Lisboa.

2.3.1 Documentação do Estudo

A grande extensão do estudo efectuado sobre metodologias ágeis de desenvolvimento, impossibilita a sua apresentação directamente como parte integrante deste relatório, no entanto, todo o estudo é apresentado em anexo a este documento (Apêndice A), e considera-se que a sua leitura atenta é essencial para uma ideal percepção de todos os factores em jogo no que diz respeito à adopção de uma metodologia. Devido a essa grande extensão, o mesmo estudo é apresentado de seguida de forma resumida, procurando-se dar uma ideia geral do funcionamento de cada metodologia, bem como evidenciar as suas principais características e a forma como estas representam ou não uma vantagem para o CI-FCUL. Impõe-se a identificação das características mais relevantes de cada metodologia, para que seja possível concluir sobre quais as metodologias que terão mais hipóteses de ser adoptadas, e quais as que à partida não terão grande ponto de interesse, num processo de desenvolvimento ágil para a instituição.

Assim sendo, através dos seguintes resumos referentes a cada uma, bem como através do quadro seguinte que faz o mapeamento entre as diferentes metodologias e as características pretendidas pelo CI-FCUL numa metodologia de desenvolvimento, torna-se possível um entendimento geral de todas as metodologias abordadas, sendo elas: o método de Programação Extrema, o método SCRUM, as metodologias Crystal Family, o método de Desenvolvimento Orientado a Funcionalidades, o método de Processo Racional Unificado, o método de Desenvolvimento de Sistemas Dinâmicos e o método de Desenvolvimento de Software Adaptativo:

Método de Programação Extrema (XP):

O método de Programação Extrema (XP) é uma metodologia baseada num conjunto de práticas de desenvolvimento provadas como funcionais em muitos projectos e que se baseia num desenvolvimento baseado em iterações curtas e por isso caracterizado por uma integração contínua de funcionalidades que vão sendo desenvolvidas iteração após iteração. Atribuindo papéis relativamente simples e comuns aos vários membros da equipa de desenvolvimento que define, apoia-se também na escrita de documentação durante todo o processo de desenvolvimento e num grande envolvimento do Cliente no processo de desenvolvimento da aplicação ou serviço em causa. Por essa razão todo o processo de recolha de

requisitos integrante do processo de desenvolvimento desta metodologia, é caracterizado por uma grande extensão e por um método meticuloso de recolher todas as intenções do Cliente através da escrita de cartões denominados “User Stories” e que procuram retratar típicas histórias de uso da aplicação, como forma de percepção de todas as suas funcionalidades.

Outra das principais caracteristicas desta metodologia é que todo o desenvolvimento é feito com base numa técnica de programação denominada “Programação em Pares” e que se baseia na codificação de funcionalides por parte de dois programadores que são colocados no centro da sala num único computador e que vão implementando funcionalidades com os restantes membros da equipa de desenvolvimento situados em seu torno.

Assim, o processo de desenvolvimento que esta metodologia define é constituído por seis fases principais, a fase de Exploração em que são recolhidos todos os requisitos da aplicação a desenvolver, a fase de Planeamento em que é feito o planeamento das tarefas de desenvolvimento em todo o projecto, o Período de Iterações em que as funcionalidades (ou user stories) vão sendo planeadas, desenvolvidas e integradas na aplicação geral, e as fases de Produção, Manutenção e Morte em que todo o projecto é lançado como produto final, gerido e por fim finalizado (caso haja necessidade disso).

Apresentando um conjunto de práticas, regras e definições que se enquadram bem em todas as características e pretensões do CI-FCUL, é uma metodologia que apresenta grandes hipóteses de adopção, principalmente devido à forma como define todo o processo de recolha de requisitos de forma bastante simples mas sobretudo completa.

Metodologia SCRUM:

O método ágil de desenvolvimento denominado como “SCRUM” concentra-se na forma como os vários elementos de uma equipa de desenvolvimento devem funcionar em conjunto, de forma a produzirem uma aplicação ou serviço flexível e capaz de resistir e ser eficiente num ambiente em constante mudança. Estando centrando num desenvolvimento por funcionalidades, esta metodologia oferece grande liberdade em termos de práticas obrigatórias de seguir no desenvolvimento, baseando apenas este na realização de “Sprints”, ou seja em pequenos ciclos de desenvolvimento de funcionalidades inteiramente marcados pela realização de reuniões e revisões constantes.

Assim, uma das principais características desta metodologia, é precisamente a realização intensa de reuniões (“scrums”) e a forma como todo o desenvolvimento está dependente de constante discussão e partilha de ideias, sendo essa a única prática que guia o processo, não havendo mesmo qualquer definição de formas de recolha de requisitos ou de formas de gestão das funcionalidades pretendidas no produto de desenvolvimento.

SCRUM define um processo de desenvolvimento de apenas três fases, sendo o Pré-Jogo a fase de planeamento e definição da arquitectura da aplicação ou serviço a desenvolver, enquanto a verdadeira implementação se processa toda na fase de Jogo com a realização dos tais “Sprints” de codificação de

funcionalidades e realização de reuniões e revisões das mesmas, para depois ser finalizado o processo na fase final de Pós-Jogo, caracterizada por uma integração de todas as funcionalidades desenvolvidas nos Sprints e pelo lançamento da versão final da aplicação.

Não apresentando um conjunto de práticas concordantes com os costumes e políticas do CI-FCUL e pelo facto de não definir um conjunto de práticas específicas para seguir no processo de desenvolvimento - com excepção para a realização regular de reuniões que aceleram o desenvolvimendo das várias funcionalidades -, nem tão pouco contemplar qualquer fase de escrita de documentação, esta é uma metodologia que não apresenta grandes possibilidades de adopção neste projecto.

Metodologias Crystal Family:

Como o próprio nome indica, este é um conjunto de metodologias ágeis de desenvolvimento que formam um família de métodos com diferentes propósitos de aplicação, em que a escolha do melhor método para um projecto se baseia na complexidade exigida por este. Assim, quanto mais complexo o projecto mais complexa será a metodologia escolhida (cada metodologia é definida por uma cor: branca, amarela, cor-de-laranja e vermelha, da mais leve para a mais pesada), sendo que para todos os projectos, esta família de metodologias baseia as suas práticas num desenvolvimento incremental e procura dar ênfase à comunicação e à cooperação entre elementos de equipa, enquanto se preocupa em reduzir ao mínimo de versões possíveis todo o desenvolvimento da aplicação.

A adopção de uma família de metodologias por parte de uma equipa de desenvolvimento constitui uma vantagem em termos de flexibilidade, diversidade e simultaneadade nos projectos possíveis de desenvolver. No entanto, a falta de definição de práticas obrigatórias de seguir nestas metodologias enfraquece o seu processo em termos de fiabilidade e obriga também à adopção de técnicas definidas por outras metodologias. Pode-se ainda dizer, que para projectos com o grau de complexidade como os tipicamente desenvolvidos pelo CI-FCUL, apenas a metodologia mais simples desta família se enquadra (Crystal Clear).

Em termos de processo de desenvolvimento qualquer metodologia desta família define apenas duas fases, sendo elas a fase de “Staging” na qual se faz todo o planeamento e calendarização do projecto, e a fase de “Iterações” em que é levado a cabo o próprio desenvolvimento, bem como a produção da aplicação desenvolvida.

Em todas estas metodologias, para além da falta de práticas bem definidas em todo o processo de desenvolvimento, pode-se ainda concluir sobre uma total inexistência de fases de revisão, documentação e realização de testes, bem como sobretudo identificar a total falta de especificação de formas de recolha de requisitos aquando do início do desenvolvimento de qualquer projecto. Juntando a estes factos, a definição de um número demasiado grande de papéis e responsabilidades pelos membros da equipa de desenvolvimento, aumenta as hipóteses de não adopção de uma metodologia deste família no CI-FCUL.

Método de Desenvolvimento Orientado a Funcionalidades (FDD):

Em inglês “Feature Driven Development”, o método de desenvolvimento orientado a funcionalidades é, como o próprio nome indica, uma metodologia centrada no desenvolvimento de projectos tendo como base as suas funcionalidades e o desenvolvimento iterativo destas. É assim, um método principalmente focado numa definição intensiva das fases de desenho e implementação de projectos, e que deixando de fora pormenores relacionados com a recolha exaustiva de requisitos, permite centrar todo o desenvolvimento na eficácia e na agilidade da implementação prioritizada de funcionalidades. Esse tipo de desenvolvimento iterativo de funcionalidades resulta na produção de um grande número de versões da mesma aplicação e por isso no enriquecimento constante de um protótipo e é suportado por formas de monitorização eficazes que garantem que qualquer funcionalidade é desenvolvida com o propósito e funcionamento adequado.

Outra das características principais desta metodologia, é o facto de ser tolerante a mudanças e de permitir que novas funcionalidades sejam englobadas no processo de desenvolvimento à medida que vão surgindo como necessárias ao produto final da implementação. O desenvolvimento baseado em iterações e centrado em funcionalidades permite um planeamento constante das funcionalidades que vão ser implementadas a cada versão do produto do desenvolvimento.

Assim, em termos de processo de desenvolvimento, esta metodologia é caracterizada por quatro fases distintas, em que na primeira é desenvolvido o modelo geral da aplicação, seguidamente é construída a lista de funcionalidades a implementar na aplicação, e planeado o desenvolvimento para cada funcionalidade que se deseja ver incluída na próxima versão da aplicação, para por fim ser levada a cabo o desenho e construção de cada funcionalidade.

O facto de toda a metodologia ser caracterizada por um processo de implementação muito bem definido e orientado à obtenção de resultados, bem como outros factores, como a existência de uma única equipa de desenvolvimento, e o constante envolvimento do Cliente na recolha de requisitos (que é no entanto mal definida na metodologia), formam um conjunto bastante favorável de argumentos, para que as ideias chave desta metodologia sejam aproveitadas na construção da metodologia a adoptar pelo CI-FCUL.

Método de Processo Racional Unificado (RUP):

Conhecido como RUP, o método de Processo Racional Unificado apresenta características que o distinguem largamente das restante metodologias abordadas, uma vez que é primeiramente caracterizado pela sua forte ligação à utilização de casos de uso para modelação de requisitos no início de cada projecto. Assim sendo, fazendo extenso uso de UML, este método apoia-se em esquemas deste género para um desenvolvimento orientado a objectos, baseando-se mesmo no mapeamento directo de esquemas para classes que implementam as funcionalidades representadas nestes.

Outras das características que fazem deste método um método eficaz para desenvolvimento de aplicações, é o facto de se basear também num grande controlo de custos, tempo e qualidade, de ser orientado para o documentação de todas as fases de desenvolvimento e da revisão das funcionalidades ser normalmente levada a cabo pelo próprio Cliente da aplicação.

O seu processo de desenvolvimento é então constituído por quatros fases: na "Incepção" são definidos todos os objectivos e necessidades do projecto e portanto esta é a principal fase de todo o processo, sendo recolhidos todos os requisitos sobre a forma de casos de uso que irão ditar as funcionalidades e o comportamento da aplicação; na "Elaboração" é construída a base de toda a aplicação e são calendarizados e planeados todos os passos do desenvolvimento das várias funcionalidades e versões da aplicação; na "Construção" são implementadas todas as funcionalidades da aplicação, bem como são feitos os testes necessários para verificação da alta qualidade de todo o desenvolvimento, e para controlo de custos e tempos de implementação ; por fim na "Transição" é lançada a aplicação mediante a aprovação dos seus Clientes e é escrita toda a documentação referente ao projecto.

O maior senão que parece existir na adopção desta metodologia no CI-FCUL relaciona-se com a orientação do mesmo para a utilização de ferramentas semi-automáticas de desenvolvimento, essencialmente baseadas em aspectos gráficos em detrimento das normais formas de codificação de aplicações. Também o uso quase exclusivo de UML para recolha e análise de requisitos não se adequa totalmente ao ambiente em que nos encontramos, bem como a definição de um largo conjunto de reponsabilidades e papéis na equipa de desenvolvimento. No entanto, numa perspectiva de desenvolvimento baseado exclusivamente numa ferramenta tipo gestor de conteúdos, esta poderia ser uma metodologia eficaz.

Método de Desenvolvimento de Sistemas Dinâmicos (DSDM):

Denominado frequentemente como DSDM, o método de Desenvolvimento de Sistemas Dinâmicos segue princípios de interacção activa entre utilizadores, e baseia-se por isso num desenvolvimento colaborativo de funcionalidades fornecendo à equipa de desenvolvimento o poder suficiente para tomar decisões, ao mesmo tempo que se suporta na realização contínua de testes durante o desenvolvimento, e na construção frequente de protótipos e versões de produtos.

Uma das suas principais características e que o permitem distanciar relativamente às restantes metodologias ágeis é a sua principal preocupação pelo tempo e recursos gastos no desenvolvimento, ajustando a quantidade de funcionalidades de acordo com os recursos disponíveis mais do que com as exigências de um Cliente (que no entanto guia o desenvolvimento através da prioritização das funcionalidades a desenvolver). A sua principal preocupação é a do respeito pelos recursos desenvolvendo as funcionalidades possíveis com o que é fornecido pelo "patrocinador" do projecto.

Quanto ao seu processo de desenvolvimento é definido por quatro fases distintas, na primeira fase denominada "Estudos de Viabilidade de Negócio" é percebida a forma como o projecto tem de ser

abordado em termos de possibilidades técnicas e de negócio, bem como em termos de riscos e de viabilidade de implementação. Na segunda fase e terceira fase, denominadas "Modelo Funcional de Iteração" e "Desenho e Montagem de Iteração" respectivamente, é feito o planeamento, desenvolvimento e teste das diversas funcionalidades de uma forma iterativa (primeiro é desenvolvido o protótipo geral para depois se ir enriquecendo este com funcionalidades) e são analisados os resultados destas como forma de antecipar as futuras iterações, sendo produzido código que implementa as funcionalidades, modelos de análise do projecto, listas prioritizadas de funções a implementar e documentos de revisão do desenvolvimento. Na última fase "Implantação", o sistema é lançado em produção, sendo dado treino aos seus utilizadores e produzidos os documentos de ajuda à utilização e de revisão do desenvolvimento.

Alguns dos principais problemas desta metodologia são: o facto de todo o suporte relacionado com as suas práticas ser na verdade fornecido por parte de uma empresa (ou grupo) privado, a existência de dependência de um "patrocinador" que detém poder sobre todos os custos do desenvolvimento e que pode levar ao abandono de grande parte dos projectos, devido ao controlo exigente de recursos e tempo, e a inexistência de um Gestor de Projecto na equipa de desenvolvimento. Estes factores fazem com que esta metodologia não seja adequada para adopção por parte do CI-FCUL.

Método de Desenvolvimento de Software Adaptativo (ASD):

O último método de desenvolvimento abordado no estudo, e reconhecido como ASD, é o metodo de Desenvolvimento de Software Adaptativo, e destaca-se dos restantes devido ao seu processo de desenvolvimento ser orientado a objectivos e não a funcionalidades. Tipicamente baseado em ciclos iterativos de desenvolvimento e no desenvolvimento rápido de aplicações, apoia-se numa constante prototipagem de aplicações e numa obtenção rápida dos resultados definidos como metas aquando do início dos projectos.

Para além do desenvolvimento iterativo, esta metodologia baseia-se na orientação ao risco, com o desenvolvimento de componentes mais críticas a ser levado a cabo em primeiro lugar, na imposição de limites de tempo sobre o desenvolvimento de cada componente, e na realização de revisões de componentes orientadas ao Cliente, para que mesmo em ambientes em constante mudança a aplicação satisfaça os objectivos deste.

O principal objectivo desta metodologia é então o de fornecer guias de desenvolvimento adequados aos desenvolvedores, para a implementação de aplicações que estão sujeitas a grandes mudanças nas suas especificações e funcionalidades. Também por esta razão todo o processo de desenvolvimento definido por esta metodologia se baseia na especulação e não no planeamento. Assim sendo, apresenta tipicamente três fases distintas de desenvolvimento: a primeira denominada "Especulação" que compreende a definição da missão e dos objectivos do projecto e construção de um calendário geral do projecto e da sua implementação, a segunda denominada "Colaboração" que se refere ao desenvolvimento propriamente dito das várias componentes que terão de integrar o produto final e que estão naturalmente relacionados

com os vários objectivos definidos na primeira fase e a terceira fase, denominada “Aprendizagem”, que compreende mecanismos de revisão das componentes desenvolvidas, para que estas possam ser refinadas de acordo com as várias características em constante mudança e para que o produto final de desenvolvimento seja uma aplicação consistente.

Existem algumas características desta metodologia que constituem pontos negativos no âmbito da adopção da mesma por parte do CI-FCUL, no entanto, o facto mais impeditivo para que isto aconteça, é o facto de ser demasiado perigoso guiar o desenvolvimento de uma aplicação pelos objectivos do Cliente da mesma, uma vez que, na maior parte das vezes, os Clientes não sabem ao certo o que pretendem ver implementado numa aplicação, tal como sobretudo, não têm consciência do esforço necessário para o seu desenvolvimento. Assim sendo, esta não parece ser a metodologia mais adequada para adopção neste projecto, nem parece contribuir com grandes práticas para o processo de desenvolvimento de aplicações pensado.

Mapeamento entre Metodologias Abordadas e Características Pretendidas pelo CI-FCUL

Características pretendidas pelo CI-FCUL	Metodologias Abordadas:						
	XP	SCRUM	Crystal Family	FDD	RUP	DSDM	ASD
Curtos ciclos iterativos de desenvolvimento de funcionalidades.	Sim	Sim	Sim	Sim	Sim	Sim	Não
Enriquecimento gradual de um protótipo de aplicação.	Sim	Sim	Não	Sim	Sim	Sim	Sim
Equipa de desenvolvimento entre 2 e 10 pessoas situadas no mesmo espaço	Sim	Sim	Não	Sim	Sim	Não	Não
Processo de desenvolvimento orientado ao Cliente	Sim	Não	Indef.	Sim	Sim	Sim	Sim
Processo de recolha de requisitos muito completo	Sim	Não	Não	Não	Não	Não	Não
Processo de desenvolvimento orientado para uma boa documentação de cada projecto desenvolvido	Sim	Não	Não	Sim	Sim	Sim	Não
Existência de mecanismos mínimos de controlo de tempo e custos do projecto, através essencialmente de reuniões, revisões ou monitorização	Sim	Sim	Não	Sim	Sim	Sim	Sim
Papéis e responsabilidades relativamente simples e eficazes.	Sim	Sim	Não	Não	Não	Não	Sim
Semanas de trabalho com cerca de 40 horas, equivalendo a 8 horas por dia por trabalhador.	Sim	Indef.	Indef.	Sim	Sim	Indef.	Indef.

Tabela 3: Mapeamento entre características pretendidas pelo CI-FCUL e metodologias de desenvolvimento aboradas

De toda a análise feita às diferentes metodologias ágeis de desenvolvimento abordadas neste projecto, conclui-se sobre a possibilidade de adopção de duas delas como método de desenvolvimento de aplicações no CI-FCUL. É de referir ainda que, para confirmação das probabilidades de adopção de cada metodologia através do quadro acima, bastará perceber o número de características requeridas para adopção no CI-FCUL que cada metodologia favorece. Observando desta forma, e juntando resumos ao

quadro acima, sobressaltam das duas formas de análise, a metodologia de Programação Extrema e o método de Desenvolvimento Orientado a Funcionalidades, como as principais metodologias a considerar na adopção de uma metodologia ágil de desenvolvimento por parte do CI-FCUL.

No entanto, e sendo que nenhuma das duas metodologias destacadas acima, por si só, parece indicada à utilização por parte deste órgão de desenvolvimento (uma vez que os pontos negativos não são ultrapassáveis), irei considerar uma junção das suas características na definição de uma nova metodologia de desenvolvimento, orientada às necessidades do CI-FCUL.

2.4 Considerações Finais

A adopção de uma metodologia de desenvolvimento por parte do Centro de Informática da FCUL é um dos principais pontos de foco de todo este projecto, uma vez que sem uma metodologia eficaz de trabalho, nenhum grupo de desenvolvimento consegue um desenvolvimento eficaz de serviços e aplicações ou uma gestão adequada dos mesmos.

No caso do CI-FCUL a grande inexistência de processos e o facto de todo o desenvolvimento ser orientado para a satisfação dos Clientes e utilizadores dos vários Serviços (uma vez que a utilização dos mesmos disso depende) resulta no interesse de adopção de uma metodologia de desenvolvimento que permita bastante flexibilidade e que seja sobretudo voltada para a obtenção de resultados capazes de satisfazer todas as partes envolvidas. Por essa razão, para o CI-FCUL interessa a adopção de uma metodologia ágil de desenvolvimento que possibilite um desenvolvimento eficaz, em detrimento da adopção da clássica metodologia de desenvolvimento "em cascata" em que todos os processos estão muito bem definidos e não são permitidas grandes alteração ou desvios no processo de desenvolvimento. A identificação da necessidade de um processo de desenvolvimento fortemente inspirado numa profunda e eficaz recolha de requisitos, motiva também a utilização de uma metodologia orientada à satisfação dos objectivos dos Clientes e ao desenvolvimento orientado a funcionalidades, sendo para isso essencial a adopção de uma metodologia cujo processo de desenvolvimento é baseado em ciclos iterativos de implementação de funcionalidades, e em que o planeamento a implementação e a integração fazem parte de cada iteração e não apenas de uma implementação geral da aplicação (no modelo "cascata" o planeamento é apenas feito uma vez, aquando do início do projecto, e não ao nível de cada implementação de funcionalidade).

Por esta razão a adopção de uma metodologia ágil de desenvolvimento baseada em indivíduos e nas suas interacções, bem como no uso de ferramentas altamente funcionais, na colaboração flexível com Clientes, e na capacidade de resposta a mudanças é uma realidade no CI-FCUL. Tal como é uma exigência, a constituição de um processo de desenvolvimento modular e iterativo apoiado num rápido feedback por parte dos Clientes, e que apresente processos de controlo de tempo de desenvolvimento das diversas funcionalidades, permitindo de forma simultânea uma implementação eficaz e documentada.

Após um estudo elaborado de diversas metodologias ágeis de desenvolvimento, interessa evidenciar uma clara vantagem de duas metodologias que se enquadram nas necessidades do CI-FCUL e que apresentam práticas possíveis de implementar no seio do grupo de desenvolvimento que forma este grupo. Tanto o método de Programação Extrema (XP), como o Desenvolvimento Orientado a Funcionalidades (FDD) apresentam características muito interessantes e que podem possibilitar uma clara melhoria no desenvolvimento de serviços e aplicações por parte do CI-FCUL caso sejam adoptados. No entanto, a adopção isolada de apenas uma destas metodologias e de todas as suas práticas, regras e restrições, traduzar-se-ia na instauração de um processo de desenvolvimento inadequado ao CI-FCUL, uma vez que nenhuma das duas apresenta argumentos totalmente concordantes com as necessidades do grupo.

Desta forma, é objectivo do capítulo que se segue, a definição de uma metodologia ágil de desenvolvimento baseada nestas duas metodologias e nas suas práticas, por forma a adequar as mesmas ás necessidades e ás pretenções de desenvolvimento do CI-FCUL.

Capítulo 3 Definição de Uma Metodologia de Desenvolvimento

Após toda a análise às diferentes metodologias estudadas, retira-se que as razões essenciais para a adopção de uma metodologia como metodologia de trabalho no CI, são, a existência de um processo de desenvolvimento acima de tudo simples e que seja em segundo lugar, bem orientado ao desenvolvimento centrado em funcionalidades identificadas por um cliente. De todas as metodologias anteriores, ressaltam as possibilidades de adopção de uma metodologia principal (e a primeira a ser definida) – Metodologia de Programação Extrema (XP) -, bem como de uma secundária - Metodologia Orientada a Funcionalidades (FDD). Estas duas metodologias, por se englobarem nas características pretendidas no CI-FCUL, são por as que constituem de forma maioritária, a metodologia proposta para adopção no Centro de Informática da FCUL. De seguida passo a explicar essa metodologia:

3.1 Processo:

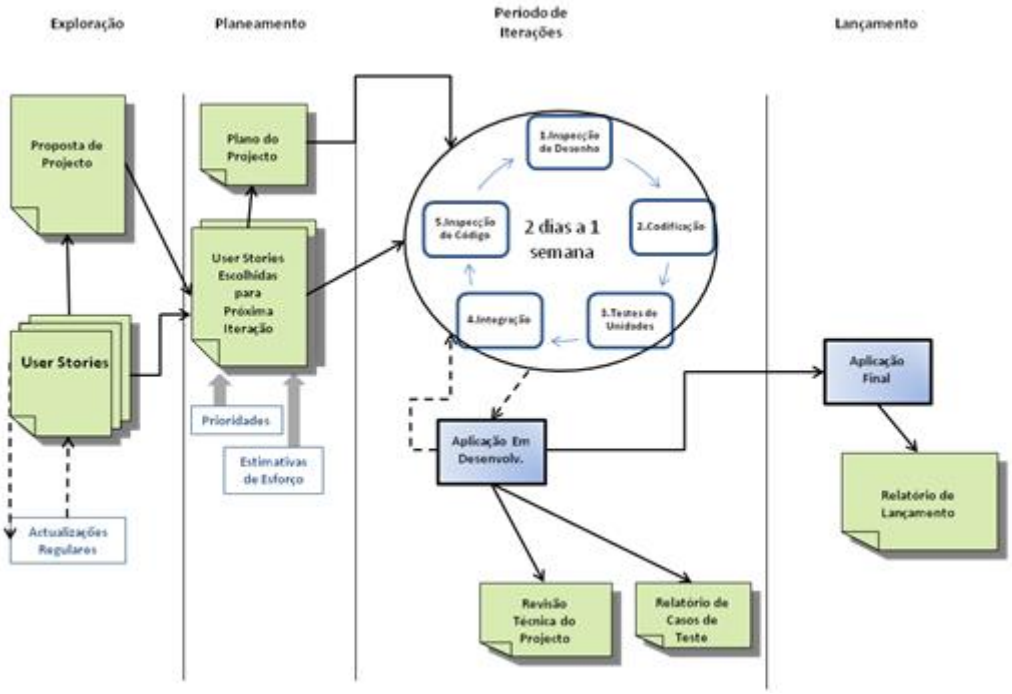


Figura 5: Processo da metodologia resultante para adopção

O processo de desenvolvimento que define a metodologia a adoptar (Figura 5), é assim totalmente inspirado no método de Programação Extrema (XP), sendo que no entanto uma das suas quatro fases, apresenta alterações relativamente a este método, inspirando-se na metodologia de Desenvolvimento Orientado a Funcionalidades (FDD) no que respeita à fase de desenvolvimento iterativo. Assim sendo, interessa descrever de forma mais pormenorizada cada uma das fases:

3.1.1 Exploração

A primeira fase, denominada "Exploração", é provavelmente a fase mais decisiva de todo o processo, uma vez que é nesta fase que se definem todos os requisitos para o projecto, e portanto é nesta fase, que todo o projecto se começa a centar nas intenções do Cliente e definir o que será a aplicação produto do desenvolvimento. Sendo práticamente "retirada" do método XP, baseia-se assim na recolha de requisitos iniciais para o projecto, sendo o seu principal objectivo a construção de requisitos em colaboração directa com o cliente. Para esse fim são utilizados "User Stories" (descritos pormenorizadamente mais à frente) que retratam de forma relativamente simples, as intenções dos utilizadores e permitem uma relação directa destas com funcionalidades específicas a desenvolver na aplicação (o processo de recolha dos "User Stories" está também descrito mais à frente aquando da descrição dos vários documentos gerados em todo o processo de desenvolvimento). É também com base nessas histórias de utilização que é produzido o documento que identifica a proposta de projecto em todos os seus objectivos e funcionalidades ("Proposta de Projecto").

Assim sendo, e resumindo esta fase a um conjunto de instruções temos o seguinte procedimento:

1. Realização de uma reunião de arranque de projecto, em que se reunem 1 ou 2 representantes da equipa de desenvolvimento, e os representantes do cliente (tipicamente também 1 ou 2). Nesta reunião de arranque é tipicamente realizado todo o processo de recolha de requisitos através da escrita de "User Stories" por parte dos vários envolvidos.

    a. A lista de "histórias" escritas passa a ser a principal fonte de desenvolvimento de todo o projecto. Como tal, sempre que são propostas novas funcionalidades por parte dos Clientes (que não foram por exemplo identificadas aquando da reunião de arranque) esta lista é actualizada e é identificada a prioridade das novas funcionalidades propostas, mantendo-se assim uma lista prioritizada de requisitos.

    b. A forma como a lista é prioritizada está sempre relacionada com o esforço de desenvolvimento identificado para cada funcionalidade e com o nível de importância que o Cliente lhe atribui no produto final. É consoante estas medidas que novas funcionalidades poder-se-ão ou não sobrepor às que foram identificadas à priori, e serem desenvolvidas primeiramente.

2. Escrita da proposta do projecto com base nas ideias retiradas da reunião da arranque e da respectiva recolha de requisitos. O conteúdo da proposta de projecto é identificado mais abaixo neste relatório, mas interessa referir que este documento é uma forma essencial de registo de um projecto e que qualquer etapa de implementação no processo de desenvolvimento, não deve ser iniciada antes da escrita deste documento.

    a. Quando é apresentado um documento descritivo de um projecto ao CI-FCUL, antes mesmo da realização da reunião de arranque, então o conteúdo desse documento é também considerado aquando da escrita da proposta de projecto, tendo-se no entanto em conta, que as conclusões retiradas da recolha de requisitos, sobrepõem-se sempre às originalmente escritas pelo cliente num documento deste género.

3.1.2 Planeamento

Também integralmente "retirada" do método XP, a segunda fase denominada "Planeamento", baseia-se na prioritização de todas as funcionalidades recolhidas em "User Stories", de acordo com pesos dados pelo cliente, e de acordo com estimativas de esforço de desenvolvimento identificadas pelo grupo de desenvolvimento. É ainda nesta fase que é elaborado todo o "Plano do Projecto", documento em que estão incluídos pormenores de calendarização de todo o projecto, estimativas de alocação de recursos, identificação de relações entre requisitos e pormenores técnicos das plataformas de desenvolvimento e pormenores de desenho como diagramas de fluxos de funcionalidades.

Assim sendo, e resumindo esta fase a um conjunto de instruções temos o seguinte procedimento:

1. Consulta ou reunião com o cliente para determinação dos requisitos a preencher para o lançamento das várias versões da aplicação. O Cliente é encarregado de especificar quais os "user stories" que deseja ver em cada versão, e de especificar qual o tempo máximo de desenvolvimento dessas mesmas versões.

    a. Toda esta definição de versões e das diferentes funcionalidades requeridas para cada uma, pode ser levada a cabo na primeira fase do projecto, aquando da primeira reunião de arranque com os clientes e logo após a escrita dos diversos "user stories".

    b. Pode ainda ser levada a cabo de forma incluída no ciclo iterativo de desenvolvimento, caso haja disponibilidade do Cliente para isso. Assim, no início de cada versão haveria uma planeamento dessa versão - com possível inclusão de novas funcionalidades e "user stories" respectivos, não pensados a início-, e uma nova reunião com o Cliente para esse mesmo planeamento. Isto permitiria deter um maior controlo sobre todo o desenvolvimento, tal como conferiria uma maior agilidade a toda a metodologia, não havendo obrigação de seguir uma linha totalmente definida de implementação do início ao fim do projecto, possibilitando-se como já foi dito, a inclusão de novas funcionalidades ou de alterações a funcionalidades já existentes, e que a início não eram

consideradas, tolerando-se desta forma alguma "mudança" nas especificações do projecto durante a sua implementação.

2. Com base na especificação anterior, é elaborado o documento do "Plano do Projecto". O conteúdo do plano do projecto, é identificado mais abaixo neste relatório, mas interessa adiantar que para além de compreender pormenores relacionados com as diferentes versões a lançar, especifica ainda a calendarização das diferentes tarefas, fazendo a divisão das memsas por recursos e por membros da equipa de desenvolvimento (a forma como todo o conteúdo é definido está também especificado mais abaixo no documento aquando da descrição do próprio documento de "Plano de Projecto").

3. Reunião com vários membros da equipa de desenvolvimento (tipicamente participam os desenvolvedores mais experientes) para o cálculo do tempo de desenvolvimento sobre os vários "user stories" requeridos para a próxima versão do produto. Tipicamente, é feito um somatório das horas de esforço relativas a cada "user story" já identificadas na primeira fase e é feita uma comparação com o máximo de tempo de desenvolvimento identificado pelo Cliente para a versão em causa (de realçar que o máximo tempo de desenvolvimento terá de ser multiplicado pelo número de membros da equipa de desenvolvimento que vão participar no projecto). Caso o tempo total de implementação se sobreponha ao somatório de esforço calculado, então terão de ser excluídos da versão as funcionalidades cujos "user stories" têm uma prioridade mais baixa (prioridade esta também identificada na primeira fase do processo da metodologia).

    a. Caso hajam dúvidas relacionadas com a prioridade de alguns "user stories" em relação a outros (por terem segundo o Cliente o mesmo grau de importância, por exemplo) cabe sempre à equipa de desenvolvimento, determinar quais os requisitos a serem satisfeitos na versão em produção, sendo essa escolha feita normalmente no sentido desenvolver as funcionalidades mais gerais e mais simples sugeridas pelos "user stories".

3.1.3 Período de Iterações

A terceira fase, dá pelo nome de "Período de Iterações" e é inteiramente baseada no ciclo iterativo de desenvolvimento da metodologia orientada a funcionalidades (FDD). A razão porque é adoptado este tipo de desenvolvimento em detrimento da respectiva fase iterativa presente no método de Programação Extrema (XP), prende-se com o facto de a técnica de "programação em pares" não se mostrar adequada para o desenvolvimento de serviços nesta organização (como é explicado na análise desse método). Desta forma, com base na prioritização de requisitos realizada nas duas fases anteriores, inicia-se o desenvolvimento iterativo das várias versões da aplicação, definindo-se com base no tempo de desenvolvimento e nos recursos, as funcionalidades a estarem presentes em cada incrementação. Nesta fase são desenvolvidos também os necessários testes sobre as funcionalidades em desenvolvimento, procurando-se o envolvimento do Cliente nestes sempre que possível.

Explicando mais ao pormenor as cinco fases do ciclo iterativo temos:

1. **Inspecção de Desenho**: O primeiro passo do ciclo iterativo, recebe da fase de planeamento o documento de "Plano de Projecto" e os "user stories" devidamente prioritizados e com base na "inspecção" destes, permite a um desenvolvedor adquirir conhecimentos necessários para a implementação de uma funcionalidade (que o "user story" refere). Nesta fase dá-se então a consulta do planeamento do projecto e dos respectivos diagramas de sequência, de classes e as formas de estruturação da funcionalidade a desenvolver, para que haja uma clara ideia do que se pretende atingir com esta antes de iniciar a sua codificação.

2. **Codificação**: Segundo passo do ciclo iterativo de desenvolvimento que consiste na implementação, por parte do(s) membro(s) da equipa de desenvolvimento, da funcionalidade "investigada" no primeiro passo de inspecção de desenho, utilizando para isso as ferramentas apropriadas (ferramentas estas que são extensamente analisadas neste documento).

3. **Testes de Unidades:** Terceiro passo do "Período de Iteracções", que consiste na realização de testes (tipicamente, uma sequência de instruções que visa verificar a alteração ou criação de dados na aplicação) sobre a funcionalidade implementada no passo anterior, envolvendo o Cliente da aplicação e finalizando-se com a aceitação do funcionamento da funcionalidade por parte deste.

4. **Integração:** Após aprovação nos testes, a funcionalidade é integrada no protótipo da aplicação, que vai sendo enriquecida até ser lançada sobre a forma de versões. Isto é feito naturalmente através das ferramentas de implementação e está por vezes englobado no passo anterior de testes de unidade.

5. **Inspecção de Código:** Último passo desde ciclo iterativo, a inspecção de código é feita quase de forma simultânea com a integração, e tem como objectivo a clarificação de padrões de código e generalização de funções com o objectivo de reaproveitamento de funcionalidades ou formas de implementação destas. Neste último passo verifica-se que a funcionalidade desenvolvida está de acordo com as especificações e requisitos iniciais, bem como com os padrões e regras de codificação que possam existir ao nível da organização ou da ferramenta de desenvolvimento.

Para além destas 5 fases, temos ainda a realização regular de reuniões de progresso e planeamento durante este ciclo iterativo, com o objectivo de garantir o controlo sobre o desenvolvimento geral e satisfazer as constantes alterações possíveis de se darem no decurso da implementação de uma aplicação ou serviço, seja como resposta a uma alteração dos objectivos da mesma para os Clientes, ou pela adição ou remoção de uma funcionalidade. Assim sendo:

6. **Reunião de Progresso e Planeamento:** Membros da equipa de desenvolvimento reunem-se entre si, podendo-se reunir também com o Cliente (caso exista disponibilidade por parte deste), para que seja feito um ponto de situação de todo o desenvolvimento. Estas reuniões

são normalmente realizadas de forma semanal, ou após a implementação de 5 funcionalidades novas na aplicação ou serviço em desenvolvimento. É nestas reuniões que é "compilado" e redigido o documento de "Revisão Técnica do Projecto", no qual são guardados: o estado de implementação das diferentes funcionalidades, as observações resultantes do desenvolvimento das mesmas e que interessam guardar para o futuro (como forma de prevenção de problemas num projecto futuro por exemplo), as novas funcionalidades introduzidas no desenvolvimento (e razão pela qual foram introduzidas), e um relatório dos casos de teste já realizados, como toda a informação referente à opinião do Cliente e ao resultado final dos mesmos (mais informação referente ao documento é dada mais à frente na descrição do mesmo).

Assim, tipicamente um membro da equipa de desenvolvimento aquando do ciclo iterativo de desenvolvimento, consulta num documento toda a informação referente à funcionalidade que vai desenvolver em seguida, codifica essa funcionalidade, testa a mesma junto do cliente procurando a sua aprovação, e por fim integra-a com o restante conjunto já implementado na aplicaçao geral e confirma que tudo obedece a critérios e regras de programação definidas pela equipa de desenvolvimento. Por fim aquando da reunião com restantes elementos da equipa de desenvolvimento, este reporta pormenores relevantes da funcionalidade.

É com base em todo este processo, que vários membros da equipa de desenvolvimento vão de forma simultânea enriquecendo e desenvolvendo uma aplicação, terminando apenas o ciclo iterativo, quando todas as funcionalidades identificadas à partida tiverem sido desenvolvidas (ou canceladas), e após aprovação do Cliente (caso seja mandatório para o projecto a total aprovação da versão final pelo Cliente).

3.1.4 Lançamento

A quarta e última fase, é mais uma vez, inspirada na fase final do processo de desenvolvimento de Programação Extrema, sendo no entanto uma conjunção das três últimas fases deste ("Produção", "Manutenção" e "Morte") numa única fase denominada "Lançamento". Baseia-se assim no lançamento para produção da aplicação ou serviço desenvolvido, fazendo-se este acompanhar pela escrita de uma relatório de lançamento normalmente composto por relatos de desenvolvimento, por guias de utilização e por outras informações relevantes para a conclusão de um projecto e respectiva manutenção.

Assim sendo, esta fase resume-se ao seguinte conjunto de instruções:

1. Lançamento da aplicação, ou entrega da aplicação ao Cliente para uso da mesma.

2. Realização de reunião de lançamento da aplicação com o objectivo de explicar todo o funcionamento da mesma ao Cliente e Utilizadores que a irão usar. Participam na reunião elementos chave da equipa de desenvolvimento, Clientes e Utilizadores típicos da aplicação.

3. Escrita do documento “Relatório de Lançamento”, que reune informação referente a funcionalidades implementadas ou canceladas, bem como considerações sobre todo o desenvolvimento levado a cabo (considerações de testes realizados, considerações do Cliente e considerações sobre o futuro), e a escrita de um guia de utilização com explicações básicas sobre a aplicação e exemplos de utilização da mesma.

3.2 Papéis Definidos na Metodologia Adoptada:

Os papéis a atribuir aos diferentes membros da equipa de desenvolvimento, em qualquer projecto desenvolvido sobre esta nova metodologia são basicamente os mesmos que na metodologia de Programação Extrema (XP). Tal como nesse método, existirão três papéis principais – Gestor, Programador e Cliente -, e estes por sua vez multiplicar-se-ão em mais alguns papéis mais específicos – Treinador, “Tester” e “Tracker” ***[Williams, 2007]***:

Gestor: Para além de ser a individualidade com maior poder de decisão em toda a equipa de desenvolvimento, tem como principais funções, a formação das equipas para cada projecto a desenvolver, tal como é responsável pela obtenção dos recursos e pela comunicação de toda a equipa de desenvolvimento com o exterior. Nesta metodologia terá grande peso na interacção com o Cliente aquando do arranque de qualquer projecto (recolha de requisitos).

Programador: Papel que se pretende como mais comum para toda a equipa do Centro de Informática da FCUL, sendo que qualquer programador é responsável por escrever código, desenhos e testes para implementação das mais variadas funcionalidades, bem como tem papel activo na identificação e estimação das diversas “User Stories” escritas pelo Cliente (em projectos em que a equipa de desenvolvimento desempenha o papel de Cliente, são os programadores os responsáveis pela escrita dos “User Stories”).

Treinador: Responsável por ensinar todo o processo de desenvolvimento a outro(s) elemento(s) da equipa de desenvolvimento. Tem ainda responsabilidades na resolução de problemas relacionados com a metodologia, bem como na monitorização de todo o processo. Este papel será tipicamente desempenhado por um Programador mais experiente e sobre um Programador sem grandes conhecimentos nas ferramentas de desenvolvimento.

“Tester”: Como o nome indica, é o programador responsável por ajudar os Clientes na escrita dos testes a cada funcionalidade, e que guia estes na sua demonstração. Será desempenhado, tipicamente para cada caso, pelo Programador responsável pelo desenvolvimento das funcionalidades a serem testadas.

“Tracker”: Individualidade que recolhe “User Stories” e informação resultante dos testes de aceitação para criação de gráficos e diagnósticos de progresso do projecto em desenvolvimento. Será tipicamente o Gestor de Projecto ou um dos seus principais Programadores.

Cliente: Normalmente é aquele que submete o projecto para realização e que detém uma ideia geral do seu propósito e funcionamento (embora na realidade isto acabe por não se verificar, tendo este apenas uma breve ideia do que quer, sendo esclarecido à medida que a aplicação é desenvolvida). Em todo o caso, na metodologia a adoptar, é sempre o Cliente que escreve os "User Stories" no processo de recolha de requisitos que marca o início de qualquer projecto. Será também o Cliente, o responsável pela prioritização das funcionalidades a implementar no projecto, e por isso, aquele que define o conjunto de "User Stories" a ser implementados em cada versão da aplicação desenvolvida. Este papel será tipicamente desempenhado por um Professor ou um Funcionário, mas poderá também ser levado a cabo pelo próprio Centro de Informática (e pela sua equipa de desenvolvimento), quando o propósito do projecto surge internamente (neste caso, todo o processo de requisitos será naturalmente mais facilitado).

No quadro seguinte, identifica-se a relação entre as diferentes fases do processo de desenvolvimento e os diferentes papéis acabados de definir:

<table>
<tr><th>Papel / Fase do Processo</th><th>Exploração</th><th>Planeamento</th><th>Desenvolvimento</th><th>Lançamento</th></tr>
<tr><td rowspan="5">Gestor</td><td>Interacção com Cliente</td><td>Atribuição de Recursos a Tarefas</td><td rowspan="5">Implementação de Funcionalidades</td><td rowspan="5">Entrega da Aplicação ao Cliente</td></tr>
<tr><td rowspan="2">Recolha de “User Stories”</td><td>Calendarização de Tarefas</td></tr>
<tr><td>Planeamento Técnico</td></tr>
<tr><td rowspan="2">Estimação de Esforço</td><td>Prioritização do Desenvolvimento</td></tr>
<tr><td>Planeamento de Versões</td></tr>
<tr><td>Programador</td><td>Estimação de Esforço</td><td>Escrita de Casos de Uso
Planeamento Técnico</td><td>Implementação de Funcionalidades</td><td>-</td></tr>
<tr><td>Treinador</td><td>-</td><td>-</td><td>Ensina Ferramentas de Desenvolvimento
Ensina Processos de Desenvolvimento</td><td>-</td></tr>
<tr><td>“Tester”</td><td>-</td><td>Planeamento de Testes</td><td>Escrita e Realização de Testes</td><td>Escreve Documentos de Suporte à Utilização (Funcionais)</td></tr>
<tr><td rowspan="2">“Tracker”</td><td rowspan="2">Recolha de “User Stories”</td><td>Planeamento de Revisões</td><td>Escreve Diagnósticos de Progresso</td><td rowspan="2">Escreve Documentos de Suporte à Utilização (Processuais)</td></tr>
<tr><td>Gestão de Riscos</td><td>Gestão de Riscos</td></tr>
<tr><td rowspan="2">Cliente</td><td>Escreve “User Stories”</td><td rowspan="2">Define Prioridades</td><td rowspan="2">Teste/Revisão de Funcionalidades</td><td rowspan="2">-</td></tr>
<tr><td>Define Prioridades</td></tr>
</table>

Tabela 4:Relação entre Fases do Processo de Desenvolvimento e os Papéis dos Trabalhadores

No quadro (tabela 4), é possível verificar as diferentes tarefas desempenhadas por cada papel definido no desenvolvimento, sendo principalmente evidente o total envolvimento do gestor em todas as fases do projecto, o envolvimento do programador apenas em tarefas relacionadas com a programação e com a definição de funcionalidades, as claras tarefas de testes, revisão e formação em que “Tester”, “Tracker” e Treinador estão envolvidos, o envolvimento do cliente durante todo o processo em tarefas relacionadas com a definição de prioridades no desenvolvimento e com a avaliação das mesmas, e por fim, o papel desempenhado por "Tester" e "Tracker" na escrita da documentação (funcional e processual) referente a todo o projecto.

3.3 Documentos Gerados com a Metodologia Adoptada:

De seguida, são especificados os processos de recolha e escrita dos documentos gerados durante todo o processo da metodologia adoptada, ou seja, são explicados na sua estrutura os documentos que servem de suporte ao desenvolvimento de qualquer aplicação ou serviço, por parte do Centro de Informática da FCUL, bem como são explicadas as circunstâncias em que cada documento é redigido.

É importante referir que nos cinco documentos a serem apresentados, quatro contam com uma estrutura identificativa semelhante que permite a qualquer altura a percepção do projecto e do documento a quem o lê. Apenas os cartões de “User Stories”, não contam com essa estrutura uma vez que são apresentados aos clientes para preenchimento e mais tarde integrados no documento de proposta do projecto. Como tal, antes mesmo de especificar cada documento apresento a forma de apresentação que todos partilham:

1. **Folha de Rosto Identificativa**
    a. Nome do Projecto
    b. Nome do Documento
    c. Informação referente à versão, designação, autor, ano, data de criação e revisões do documento em causa.
2. **Espaço de Identificação do Projecto** (Colocado no início de cada documento)
    a. Nome do Projecto
    b. Enquadramento do Projecto
    c. Sigla do Projecto
    d. Ano do Projecto
3. **Cabeçalho Identificativo de Versão do Documento**
    a. Informação da versão, designação, autor e ano do documento (já presente na folha de rosto) é repetida por cada uma das páginas de um documento.
4. **Rodapé Identificativo do Projecto**

a. Informação referente à instituição em que o projecto se realiza bem como nome e data do projecto, é repetida por cada uma das páginas de um documento.

De seguida, aborda-se cada um dos documentos de forma individual:

3.3.1 User Stories

Definição: Em português significa "História de Utilizador" e baseia-se numa forma de definir requisitos ou necessidades de um cliente sobre a forma de um pequeno conjunto de pequenas frases. É uma forma de compromisso entre um Cliente e a equipa de desenvolvimento, onde são prioritizados requisitos na perspectiva de importância para o primeiro, e avaliados em esforço de desenvolvimento na perspectiva dos segundos ***[Williams, 2007]***. Assim, com cerca de 60 a 100 elementos deste tipo é possível descrever, prioritizar e desenvolver um serviço centrado nas necessidades de um cliente tendo em conta os custos de todo o processo de desenvolvimento ***[Khramtchenko, 2004].***

Processo de Recolha de Requisitos usando "User Stories": ***[Kumar, 2003; Kumar, 2009]***

1. Equipa de Desenvolvimento reúne-se com Cliente para o processo de recolha de requisitos.
    a. Cada tipo de utilizador recebe um conjunto de cartões e escreve as "User Stories" correspondentes às funcionalidades que deseja ver implementadas no projecto.
        i. Cada "User Story", contém **"Título"**, **"Prioridade"** (valor 1 a 5 ou qualitativo), **"Estimativa de Esforço"** (valor tipicamente em dias) e **"Descrição"**.
        ii. Cada "User Story" deve ser descrita em cerca de três frases curtas.
        iii. Na descrição devem ser usados pequenos conjuntos de palavras de forma a intuir o Cliente daquilo que se pretende de si ("No Papel de…", "Quero que…", "Para que…").
        iv. É importante, cada "User Story" conter também um "**Id"** (que a identifique univocamente)
        v. Devem ainda ser guardados "**Comentários"** (que podem ser pormenores da recolha ou especificações directas da funcionalidade descrita na "User Story") e "**Nome"** de quem escreveu cada uma.
        vi. Relativamente aos diferentes campos que constituem uma "User Story" é de referir que os campos "Título", "Prioridade", "Descrição", "Comentários" e "Nome" são preenchidos pelos clientes da aplicação, enquanto que os campos "Estimativa de Esforço" e "Id" são da responsabilidade do membro, ou membros (1 ou 2), da equipa de desenvolvimento.
    b. Todas as "User Stories" são guardadas num documento acessível por todas as partes envolvidas no processo de desenvolvimento.

c. Equipa de desenvolvimento decide quanto tempo de desenvolvimento existe até ao lançamento da próxima versão da aplicação.

    i. Calculam-se os pontos para atribuir aos requisitos apurados (se tempo de desenvolvimento é de X semanas com Y horas de trabalho e existem Z pessoas na equipa de desenvolvimento então existe um total de X*Y*Z pontos para atribuição (os pontos reflectem assim o número total de horas de desenvolvimento).

d. Atribuição de pontos a "User Stories" (e respectiva alocação de membros da equipa de desenvolvimento a funcionalidades).

    i. Com base nas horas de estimativa de esforço calculadas pelos desenvolvedores e na prioridade dada pelos clientes às diversas “User Stories” calculam-se quais as funcionalidades de implementação mais urgente, e quais poderão ser implementadas até ao lançamento da próxima versão.

    ii. (ou) Reunir novamente com Cliente e pedir para este fazer a atribuição dos pontos por ordem das funcionalidades mais importantes para desenvolvimento.

    iii. (ou) Reunir com equipa de desenvolvimento e fazer a atribuição dos pontos por ordem das funcionalidades que se consideram mais importantes de inclusão na próxima versão.

2. Equipa de Desenvolvimento implementa “User Stories” que foram escolhidos para o próximo lançamento da aplicação.

3. Enquanto não tiverem sido desenvolvidos todos os “User Stories” (ou respectivas funcionalidades) recolhidos a início, ou enquanto não houver satisfação completa por parte do Cliente, o processo de desenvolvimento volta sempre ao ponto 1.c.

    a. No entanto, existe uma renovação constante de “User Stories”, podendo haver inclusão de novas funcionalidades ou cancelamento de algumas ainda não desenvolvidas, a qualquer altura do projecto. Essas alterações são normalmente feitas pelo Cliente, ou exigem pelo menos consentimento da parte deste.

Estrutura: Originalmente distribuídos pelos vários representantes do cliente, os cartões de “User Stories” que permitem a recolha inicial dos requisitos de cada projecto ou serviço têm a seguinte estrutura:

Id:	Título:	Origem:
Descrição: No papel de... Quero que ... Para que...		
Comentários:		Prioridade:
		Esforço Estimado:

Figura 6:Exemplo do formato de um "User Story"

No campo "Id" é identificado o ""User Story"" de uma forma unívoca. Tipicamente um Id será formado por uma palavra referente ao projecto e por um número identificativo do documento.

No campo "Título" é identificada a funcionalidade descrita pelo "User Story", como que sumarizando o mesmo em poucas palavras.

No campo "Origem" é registado o nome/papel do Cliente que escreveu o "User Story".

No campo "Descrição" é descrito o "User Story" com o auxílio de pequenos conjuntos de palavras que têm a função de facilitar a escrita ao Cliente e o entendimento da equipa de desenvolvimento. "No papel de…" ou "Sendo um…" identifica o tipo de utilizador que vai efectuar uma operação; "Quero que…" ou "Pretendo…" descreve a operação a ser executada; "Para que…" descreve o objectivo geral da operação descrita, permitindo perceber o seu cabimento em todo o projecto.

No campo "Prioridade" o Cliente indica através de números de 1 a 5 ou através de um conjunto de qualificações – Muito Baixa, Baixa, Média, Alta, Muito Alta – o quanto importante para si é a inclusão da funcionalidade sugerida na aplicação em desenvolvimento.

No campo "Esforço Estimado" a equipa de desenvolvimento (ou o seu representante) indica em número de horas o esforço estimado para o desenvolvimento da funcionalidade sugerida pelo "User Story".

No campo "Comentários", para além de informações relativas a pormenores de recolha de requisitos, podem ser recolhidas informações que por si só desencadeiam o desenvolvimento de novas funcionalidades ou de novas variantes da descrita originalmente.

Um exemplo prático para cada uma destas propriedades é dado de seguida para o mesmo caso de uso:

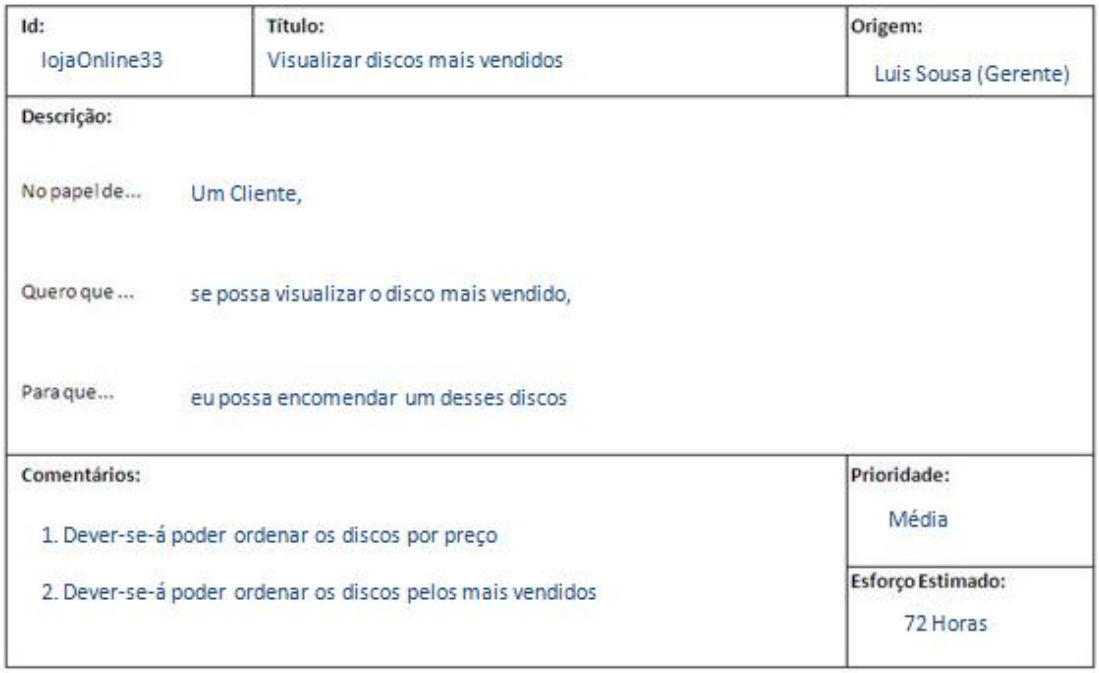


Figura 7: Exemplo do conteúdo de um "User Story"

O documento real que serve de modelo de "User Story" e que foi visualizado mais acima é incluído como anexo a este relatório no documento de "Plano do Projecto" (Apêndice B- Plano de Projecto) a este relatório.

3.3.2 Proposta de Projecto

Definição: O documento denominado como "Proposta de Projecto" é aquele que marca de forma oficial a recepção de uma "ideia" de projecto por parte do Centro de Informática vinda de um Cliente. Este documento servirá como introdução geral ao Projecto, para todos aqueles que se relacionarão com este de forma directa ou indirecta e que não participaram no processo de recolha de requisitos ou nas reuniões iniciais com o Cliente, ao mesmo tempo que servirá como suporte ao arranque do desenvolvimento sumarizando todas as funcionalidades e características do projecto a implementar.

Processo de Escrita: É concebido pelo próprio Centro de Informática, sendo construído com base nos "User Stories" escritos pelos Clientes, sendo desta forma um resumo de todos os que foram escritos, ou das funcionalidades sugeridas por estes. Quando é apresentado pelo próprio Cliente, aquando do início do projecto, um outro documento descritivo de proposta, esse documento servirá sempre como auxílio para a criação da "Proposta do Projecto", permitindo confirmar assumpções feitas sobre os "User Stories" que lhe servem de base.

Estrutura: A sua estrutura é a seguinte:

1. Espaço de Identificação do Projecto
2. **Informação Básica**
    a. Gestor do Projecto
    b. Cliente do Projecto
    c. Data de Início Estimada
    d. Data de Fim Estimada
    e. Duração Estimada do Projecto
    f. Descrição Geral do Projecto
    g. Objectivos do projecto.
3. **Requisitos**
    a. Lista de "User Stories" escritos no processo de recolha de requisitos e que representam as funcionalidades a implementar.
    b. Definição da Equipa de Desenvolvimento (e respectivas cargas horárias e tempo no projecto de cada membro integrante da equipa).
    c. Definição de Ferramentas e Recursos Tecnológicos Suplementares a utilizar na Implementação.
4. **Outros Detalhes do Projecto**
    a. Custos Suplementares do Projecto (Recursos, Ferramentas ou Licenças necessárias de aquisição).
    b. Benefícios do Projecto.
    c. Documentos Relacionados com o Projecto.

O documento real que serve de exemplo de Proposta de Projecto é fornecido como anexo (Apêndice B - Proposta de Projecto) a este relatório.

3.3.3 Plano do Projecto

Definição: Ao contrário dos restantes documentos definidos no processo de implementação de cada projecto segundo a metodologia definida, a construção de um "Plano de Projecto" abrange mais que um simples documento. Por outras palavras, são abrangidos dentro de um documento, um conjunto de vários documentos que especificam os diversos pormenores de planeamento de um projecto. É por esta razão, o documento geral sem qual um projecto não pode ser realizado de forma alguma, o mais indispensável de todos os documentos em que a metodologia se suporta, permitindo no seu conjunto, responder a perguntas como: "O que deve ser feito para atingir os objectivos do projecto?", "Como vai ser feito?", "Quem o vai fazer?" e "Quando estará feito?".

Processo de Escrita: Sendo um documento que guia todo o processo de desenvolvimento, deve ser redigido antes do arranque da implementação de funcionalidades e após a recolha de requisitos junto dos clientes, sendo redigido pelo gestor do projecto em cooperação com os desenvolvedores e o próprio cliente.

Estrutura: Na sua constituição são abrangidos os seguintes documentos secundários:

1. Espaço de Identificação do Projecto
2. **Plano de Acção**
    - **a.** Prioritização das funcionalidades a desenvolver.
    - **b.** Calendarização de Tarefas (datas iniciais e finais são estimadas).
    - **c.** Atribuição de Recursos a Tarefas (desenvolvedores e ferramentas envolvidas no desenvolvimento de cada funcionalidade)
    - **d.** Planeamento de Versões (com estimação de tempos de desenvolvimento)
    - **e.** Planeamento de Testes (com descrição do teste e data planeada para a sua realização).
    - **f.** Planeamento de Revisões
3. **Descrição Geral de Tarefas**
    - **a.** Casos de Uso Geral (Mostra relações entre actores, tarefas e sistema)
    - **b.** Casos de Uso de Funcionalidades (Especifica cada funcionalidade, em actores, requisitos e cenários sequenciais de utilização).
4. **Plano Técnico**
    - **a.** Diagramas de Classes
    - **b.** Diagramas de Sequência
    - **c.** Estruturação de Funcionalidades

5. **Gestão de Risco**

    a. Identificação de Riscos (com respectiva possibilidade de ocorrência e impacto no projecto).

    b. Mitigação dos Riscos Identificados (formas de evitar, reagir e monitorizar os riscos identificados).

Observações: Enquanto o "Plano de Acção" define todos os pormenores necessários antes de partir para a implementação de qualquer projecto, e a "Descrição Geral das Tarefas" permite aos desenvolvedores terem um documento de suporte para percepção da forma de codificação de cada funcionalidade, o "Plano Técnico" destina-se à descrição de pormenores técnicos do desenvolvimento que podem ser definidos antes do arranque deste. Os "Diagramas de Classes", "Diagramas de Sequência" e a "Estruturação de Funcionalidades", dependem pois das ferramentas utilizadas na implementação e por isso faz sentido serem definidas de forma centrada nestas (exemplo do preenchimento desta área do documento será dado mais à frente aquando da definição das ferramentas de desenvolvimento).

Definição do "Plano Técnico" e da "Estruturação de Funcionalidades"

De todos os documentos secundários referidos como fazendo parte do "Plano do Projecto", o "Plano Técnico" é aquele que se relaciona mais fortemente com as ferramentas a usar na metodologia que irá ser adoptada, e é por isso o documento que permite fazer a ligação entre ferramentas e metodologias, especificando quais os objectos necessários de implementar para representar os diversos "user stories" identificados no levantamento de requisitos, e a forma como todas as sequências de acções que estes representam são na realidade representadas ao nível da implementação.

1. Assim sendo, nos Diagramas de Classes são representados todos os objectos codificados nas ferramentas de desenvolvimento e são interligados segundo as suas relações e a sua proximidade. Ligações entre os diversos objectos representam na realidade a forma como cada funcionalidade é formada por um conjunto de objectos ou funções desenvolvidas ao nível das ferramentas. Classes interligadas são classes de alguma forma colaboram em conjunto para a realização de uma tarefa identificada pelos utilizadores da aplicação.

2. Nos Diagramas de Sequência é esclarecida a forma como todos os intervenientes de cada acção a realizar sobre o produto do desenvolvimento, funcionam em conjunto para atingir os fins determinados inicialmente. Ao olhar para um Diagrama de Sequência, um desenvolvedor ganha uma ideia de todo o funcionamento de uma funcionalidade, ficando esclarecido sobre a sequência de acções necessárias de realizar pelos vários envolvidos, para que uma funcionalidade permita de facto realizar uma tarefa identificada como necessária para os utilizadores da aplicação.

3. Por último, mas como ponto mais importante desta definição de documento, interessa referir que na "Estruturação de Funcionalidades" é feito um mapeamento directo, entre as

várias funcionalidades identificadas como requisitos para a aplicação a desenvolver, e as classes, objectos, ou o tipo de módulos, necessários de implementar ou utilizar. Para que isto seja feito é no entanto necessário, um bom conhecimento da ferramenta ou plataforma de desenvolvimento, bem como de todas as funcionalidades a desenvolver, e portanto de todos os requisitos da aplicação. Poderia ser dado um exemplo de "Estruturação de Funcionalidades" neste capítulo, no entanto e devido à sua forte ligação com as ferramentas usadas no desenvolvimento, fará apenas sentido proceder a uma especificação desta, numa secção mais à frente neste documento, após a definição das ferramentas a utilizar no desenvolvimento.

O documento real que serve de exemplo de Plano de Projecto é fornecido como anexo (Apêndice B - Plano do Projecto) a este relatório.

3.3.4 Revisão Técnica do Projecto

Definição: Tem como objectivo ir guardando o estado do desenvolvimento a cada nova implementação, guardando uma lista actualizada do estado das diferentes funcionalidades concluídas ou canceladas. Serve como forma de controlo e de monitorização de todo o desenvolvimento de uma forma relativamente simples, fazendo também o registo de todos os casos de teste efectuados a cada altura na implementação.

Processo de Escrita: Escrito durante a fase de implementação, pode ser inicialmente definida a periodicidade das actualizações, fazendo sentido que em alguns projectos as actualizações sejam feitas a cada implementação, enquanto noutros a cada versão, o ideal será quase sempre o meio-termo e após a codificação e teste de algumas funcionalidades (por forma a que o documento seja actualizado no mínimo duas vezes a cada versão).

Estrutura: A informação está guardada da seguinte forma:

1. Espaço de Identificação do Projecto
2. **Acompanhamento do Desenvolvimento**
    a. Funcionalidades Desenvolvidas, em Desenvolvimento e Canceladas (com número de horas de desenvolvimento, datas de início e de fim de codificação, e variável que indica implementação ou cancelamento).
    b. Observações de Desenvolvimento (com campo de observações que permite justificar cancelamentos ou problemas no desenvolvimento das várias funcionalidades).

c. Novas Funcionalidades para Desenvolvimento

3. **Acompanhamento de Casos de Teste**

   a. Casos de Teste Realizados (com descrição do teste, feedback do cliente e registo de aprovação do teste).

Observações: Interessa referir que a possibilidade de inserção de novas funcionalidades a meio do desenvolvimento da aplicação não pretende provocar o reordenamento de todo o desenvolvimento na prioritização das funcionalidades a desenvolver. Novas funcionalidades devem pois ser sempre desenvolvidas no fim da versão em que são inseridas ou no início da próxima, representando um esforço adicional na duração do desenvolvimento, sem que no entanto esse esforço seja significativo na duração total (isto pressupõe no entanto que o planeamento foi feito correctamente).

O ponto terceiro deste documento – "Acompanhamento de Casos de Teste" – é na verdade o documento que está representado de forma isolada no esquema do processo de desenvolvimento. Este documento pode existir de forma isolada caso seja demasiado extenso, no entanto e na maior parte das vezes, fará sempre parte do relatório técnico do projecto.

De referir ainda, que quando a periodicidade de actualização deste documento se relaciona com períodos de tempo e não com conclusões das várias versões da aplicação, então existirão sempre funcionalidades em desenvolvimento e que por isso se podem encontrar em qualquer uma das fases de desenvolvimento definidas no processo da metodologia (fase do desenvolvimento iterativo de funcionalidades). Nesses casos a indicação da fase em que se encontram (1 a 5), é essencial para a percepção do grau de completude das mesmas e para compreensão do estado geral de desenvolvimento da aplicação.

O documento real que serve de exemplo de Relatório de Revisão Técnica de Projecto é fornecido como anexo (Apêndice B - Revisão Técnica do Projecto) a este relatório.

3.3.5 Relatório de Lançamento

Definição: Por "Relatório de Lançamento" entende-se o documento necessário de elaborar aquando do lançamento da versão final da aplicação desenvolvida. Por esta razão é relevante a escrita de simples guias de utilização do serviço desenvolvido, bem como uma enumeração das funcionalidades que ficaram por implementar. Tem também como objectivo ser o único documento necessário de consultar após conclusão de cada projecto desenvolvido sobre esta metodologia, contendo assim toda a informação essencial à percepção do que foi o desenvolvimento.

Processo de Escrita: Como o próprio nome indica, este documento deve ser escrito após conclusão da codificação das funcionalidades requeridas para o projecto, e deve aproveitar toda a informação que foi registada no relatório de revisão técnica bem como verificar e ajustar alguma informação documentada no plano de projecto.

Estrutura: O "Relatório de Lançamento" contém assim a seguinte informação:

1. Espaço de Identificação do Projecto
2. **Guia de Utilização**
    a. Utilização Básica da Aplicação Desenvolvida
    b. Exemplos de Utilização
3. **Conclusões Técnicas**
    a. Lista de Funcionalidades Implementadas
    b. Lista de Funcionalidades por Implementar
    c. Considerações sobre Desenvolvimento
    d. Considerações do Cliente
    e. Possíveis Extensões Futuras ao Projecto

O documento real que serve de exemplo de Relatório de Lançamento é fornecido como anexo (Apêndice B - Revisão Técnica do Projecto) a este relatório.

3.4 Considerações Finais

A definição de uma metodologia de desenvolvimento para adopção no Centro de Informática é um ponto central em todo este projecto, e por essa razão a sua definição pormenorizada, bem como dos seus processos, é essencial para que a sua adopção por parte do grupo seja uma realidade. De todo este capítulo interessa concluir que a definição de uma metodologia de desenvolvimento baseia-se na conjunção de práticas de duas das metodologias estudadas no capítulo anterior, a Metodologia de Programação Extrema (XP) e a Metodologia de Desenvolvimento Orientado a Funcionalidades. Assim sendo, quase todas as fases do processo de desenvolvimento de aplicações e serviços são inspiradas nas fases da metodologia XP, com destaque para o levantamento de requisitos com base em "user stories" e com grande envolvimento do Cliente da aplicação. No entanto, a fase que marca o desenvolvimento iterativo das várias funcionalidades identificadas pelo levantamento de requisitos, é adoptada da segunda metodologia referida, em detrimento da implementação usando "Programação em Pares" tipicamente utilizada pelo método de Programação Extrema.

Dependente da metodologia e como real resultado desta, existem um conjunto de documentos que pretendem documentar todo o processo de desenvolvimento de qualquer projecto levado a cabo pela equipa de desenvolvimento do CI-FCUL. Neste grupo de documentos destacam-se os "User Stories" para

recolha de requisitos, e os documentos de Plano de Projecto, de Revisão Técnica de Projecto e de Lançamento. No primeiro é formulado todo o plano de acção para o desenvolvimento das funcionalidades a serem integradas na aplicação, bem como é definido uma plano técnico que descrever a forma como todas as funcionalidades se comportam e se organizam e estruturam nas ferramentas usadas no desenvolvimento. No segundo documento, é feito um acompanhamento de todo o desenvolvimento e dos casos de teste, por forma a ter uma ideia momentânea do estado de desenvolvimento da aplicação e para que fiquem registados todos os testes efectuados no desenvolvimento com as respectivas respostas dos Clientes às mesmas. Por fim, no terceiro documento principal, é feito todo um resumo do desenvolvimento, compilando uma lista de funcionalidades que foram desenvolvidas e de todas as que acabaram por não ser implementadas, especificando as razões para que tal tenha acontecido, tal como ainda são guardadas todas as considerações finais do Cliente com ideias para desenvolvimento futuro e é ainda fornecido um guia de utilização do produto desenvolvido, para ajudar os utilizadores na sua utilização.

Por fim, interessa só referir que a definição dos papéis e responsabilidades que toda a equipa de desenvolvimento terá associada, é feito de forma relativamente simples, com um número relativamente pequeno de papéis - Gestor, Programador (que pode desempenhar depois funções de Treinador, “Tester” ou “Tracker”) e Cliente – que permitem perceber por si só o tipo de tarefas que executa cada membro contemplado com uma das responsabilidades.

Até este ponto, foi definida a metodologia de desenvolvimento a adoptar no CI-FCUL, sendo levados a cabo agora nos capítulos seguintes, os processos de estudo relativamente às ferramentas que irão ser utilizadas juntamente com a metodologia definida. Tanto em termos de ferramentas de gestão dos conteúdos Web produzidos por cada projecto, como em termos de plataformas de desenvolvimento para implementação facilitada de típicas aplicações e serviços Web, é abordado um grande conjunto de ferramentas para que seja possível concluir sobre a mais ou as mais adequadas ao uso no CI-FCUL em incorporação com a metodologia ágil de desenvolvimento definida.

Capítulo 4 Análise de Ferramentas

4.1 Estudo de uma Plataforma de Gestão de Conteúdos Web

4.1.1 Introdução:

Com a generalização do fornecimento de conteúdos e serviços por toda a Web, a produção simplificada e a gestão facilitada dos mesmos torna-se uma preocupação presente para quase todas as empresas.

O Centro de Informática da Faculdade de Ciências da Universidade de Lisboa (CI-FCUL), como núcleo de produção informática que é, possui um grande conjunto de serviços e sítios Web que oferecem aos seus alunos, professores e funcionários a possibilidade de efectuar todo o tipo de operações via Internet, desde inscrições, a inquéritos, a operações de gestão dos vários sistemas de informação existentes por toda a Faculdade. Por esta razão, torna-se imperativo o uso de tecnologias que permitam simplificar a gestão de todos os serviços desenvolvidos e prestados, e que permitam separar tarefas rudimentares de simples gestão de processos de desenvolvimento mais complexos. Aliás, existem mesmo no CI-FCUL um grupo de técnicos responsáveis por tarefas relacionadas com a simples gestão de conteúdos ou desenvolvimento de serviços relativamente simples (resumem-se à apresentação de conteúdos estáticos e à revisão dos conteúdos gráficos) e um outro responsável pelo desenvolvimento de novas aplicações mais complexas com serviços dinâmicos que tratam informação dos próprios repositórios de dados (serviços dinâmicos Web como formulários, processos de Login, etc.)

Sistema de Gestão de Conteúdos - CMS

A forma mais eficaz de gerir e editar conteúdos de uma forma distribuída e colaborativa, bem como de simplificar o processo de criação de conteúdos básicos, é utilizando um Sistema de Gestão de Conteúdos ou CMS ("Content Management System"). No entanto, devido ao grande número de ferramentas deste género presentes no mundo Web nos dias de hoje, é necessário encontrar um CMS que seja adequado para o tipo de tarefas e o tipo de gestão pretendida por cada organização que adopta a sua utilização.

Actualmente, o processo de gestão de conteúdos Web, pode ser dividido em várias etapas de tratamento de informação ***[Terra, 2002]***:

1) Criação de Conteúdos

2) Revisão de Conteúdos

3) Indexação e Controlo de qualidade

4) Publicação de Conteúdo

5) Revisão Periódica de Conteúdos

6) Arquivamento/Eliminação de Conteúdos

Um CMS tem como principal objectivo a facilitação e agilização da gestão de conteúdos de sítios Web, fornecendo acesso aos conteúdos de uma organização através de uma interface única e através do simples uso de um browser e separando os conteúdos, do desenho gráfico do sítio Web que permite a apresentação desses mesmos conteúdos.

Podemos dizer assim, que é tarefa de um CMS simplificar cada uma das etapas acima identificadas, para que mesmo um utilizador sem grandes conhecimentos programáticos possa após um pequeno período de tempo, ser capaz de trabalhar com conteúdos deste género.

Funcionalidades de um CMS

Um mecanismo de gestão de conteúdos é regra geral constituído por um conjunto de módulos em cada um fornece um serviço na plataforma. São funcionalidades implementadas por estes as que a seguir passo a enumerar:

1. Gestão de interface com os utilizadores (usabilidade e arquitectura da informação)
2. Gestão de controlo de acessos dos utilizadores (autentificação, autorização e criação de perfis/tipos de utilizadores).
3. Criação, edição e armazenamento de conteúdos em diversos formatos (HTML, PHP, CSS, etc.)
4. Controlo de dados, como fluxos de acções e ciclos de vida dos documentos.
5. Indexação e mecanismos de busca de conteúdos.
6. Uso intensivo de meta dados ou de propriedades que descrevem o conteúdo.
7. Disponibilização de informação em formatos XML, e sua agregação em diferentes fontes.
8. Gestão de versionamento (controlo de versões de ficheiros de dados).
9. Registo de historial de acções executadas sobre os conteúdos e possibilidade de as desfazer para efeitos de segurança.
10. Edição de conteúdos de forma descentralizada através de qualquer browser
11. Publicação de conteúdos baseada em tempo, podendo impor um prazo de início e/ou fim da componente.

Uma descrição mais elaborada de cada uma destas funcionalidades será dada mais à frente neste relatório, no âmbito da escolha de um tipo específico de Sistema de Gestão de Conteúdos tendo em conta as necessidades do CI-FCUL.

Porquê CMS OpenSource?

A existência de um grande número de aplicações de gestão de conteúdo, a necessidade de personalização dessas aplicações (e alteração do seu código fonte por forma a reflectir uma necessidade

do grupo de desenvolvimento), o facto de estas serem utilizadas num processo de desenvolvimento que abrange ferramentas mais complexas, mas sobretudo, a possibilidade de adopção de uma ferramenta livre de custos de aquisição e suportada por uma larga comunidade de utilizadores e desenvolvedores, leva a que nos centremos na pesquisa de um CMS de licença gratuita e código aberto (OpenSource). Para além disso, este tipo de CMS dá-nos a oportunidade de abordar directamente o problema poupando recursos financeiros que poderão ser assim direccionados para o aprofundamento de conhecimentos tecnológicos da equipa de desenvolvimento. Ao mesmo tempo, permite ainda alguma flexibilidade na gestão e expansão evolutiva das aplicações, à medida que os requisitos destas se forem alterando e faz uso do conhecimento existente e de módulos de extensão que poderão ser desenvolvidos no futuro por outros utilizadores integrantes da mesma comunidade open-source.

A existência das comunidades open-source que sustentam cada aplicação são assim, como já referimos, a principal forma de suporte ao uso das mesmas. É fácil aprender, através de experiências de outros membros, as formas mais fáceis de utilização do software tal como é possível perceber, quais as funcionalidades que já se encontram completas, e quais aquelas que ainda precisam de ser trabalhadas para que possam no futuro ser incluídas no nosso CMS.

Características Pretendidas no CMS:

Havendo mesmo assim um vasto número de CMS deste tipo, para que seja possível reduzir a nossa pesquisa a um número mais limitado de possibilidades, interessa fazer um levantamento das propriedades que requeremos para uso na plataforma CMS a adoptar:

- **Apresentação flexível de conteúdos**: Sistema que possa englobar sobre a forma visual, a imagem da empresa e do serviço em desenvolvimento de forma relativamente fácil. Sistema de templates de apresentação que permita especificar pequenas variações a uma forma de apresentação geral.

- **"Workflows" simples**: Mecanismos de sistema pouco complexos no que diz respeito à aprovação de mudanças ou de novos conteúdos e à atribuição destas tarefas de revisão aos vários tipos de utilizadores da plataforma. Normalmente será um sistema de fluxo de acções de dois passos apenas: Submissão → Aprovação (Publicação).

- **Reutilização de conteúdos**: Sistema capaz de ser suficientemente modelado, para que seja possível utilizar em várias aplicações diferentes funcionalidades ou características comuns, sem que seja necessário novo esforço de desenvolvimento.

- **Utilização simples**: Plataforma fácil de utilizar, que não necessite de mais que algumas horas/dias de treino, e que permite executar as tarefas sem demasiada complexidade e de forma bastante intuitiva e natural (pouca programação e interface simples).

- **Gestão simples**: Ferramenta que possibilite uma gestão simples dos vários serviços fornecidos e desenvolvidos pelo CI-FCUL, gestão essa que possa ser feita por técnicos que

normalmente estavam habituados a gerir os mesmos serviços através de simples alterações de códigos PHP/HTML (ou mesmo por pessoal não técnico).

- **Criação de conteúdo WYSISYG**: Do inglês "What-you-see-is-what-you-get", pretende-se que a ferramenta contenha editores embebidos para a criação de conteúdo que permita ter uma visualização real do conteúdo que se está a criar ou editar.

- **Capacidade de aceder a todo o conteúdo criado pelo CMS**: A plataforma de gestão de conteúdo terá de permitir de forma relativamente simples, o acesso exterior (por outra aplicação como a DIF2.0 2.0) a qualquer conteúdo por si criado. Este acesso pode ser feito sobre a forma de URL directo, ou por via de programação através do uso de bibliotecas que incluem estes conteúdos.

Concluímos pelas propriedades enumeradas, que o tipo de CMS pretendido é na sua essência uma plataforma de gestão de conteúdos Web (WMS) com capacidades básicas de gestão de documentos e ficheiros (DMS).

Sabemos também que a procura de uma plataforma deste género, para além de ser focada nas características primeiramente indicadas, terá sempre que ser comprovada através da instalação e experimentação da própria plataforma. Isto acontece, porque mesmo sendo plataformas open-source, existe uma tendência para que nem sempre as características apresentadas à partida existam na prática, e por isso só o uso da própria ferramenta pode comprovar que esta satisfaz na realidade as nossas necessidades. No capítulo seguinte, irei enumerar, caracterizar e relatar o uso de várias ferramentas CMS abordadas durante esta pesquisa, especificando pormenores de utilização marcantes e chegando a uma conclusão sobre a sua possível adopção e uso no CI-FCUL.

4.1.2 Sistemas de Gestão de Conteúdos Abordados:

A adopção de um Sistema de Gestão de Conteúdos é sempre precedida de um estudo comparativo elaborado, que procura experimentar várias ferramentas do género e perceber qual delas é de facto capaz de responder às necessidades existentes na organização. Neste capítulo, procuro documentar esse mesmo estudo para o Centro de Informática da Faculdade de Ciências da Universidade de Lisboa.

Repositórios de CMSs

Antes de mais, convém referir que o uso e a consulta de outros sítios dedicados ao uso de CMS open-source, contribuiu para ajudar nesta pesquisa e na "descoberta" de nomes de CMS para sua posterior experimentação: através de www.CMSReview.org pude encontrar um grande número de informações relativas a centenas de gestores de conteúdos, proceder a comparações entre estes, bem como filtrar a pesquisa por ferramentas que contenham apenas certas características técnicas; e acedendo a www.opensourcecms.com foi-me possível experimentar diversos CMSs através do uso de sítios

demonstração que reflectem o uso de cada plataforma, evitando assim processos mais longos de instalação dos mesmos.

Documentação do Estudo

Sem mais demoras, passo a explicar a forma de apresentação deste extenso estudo.

O primeiro passo foi o de perceber quais os diferentes CMS open-source que podiam ser alvo de estudo e portanto considerados como hipótese de adopção por parte do Centro de Informática. Apurar as características dos diferentes CMS face às necessidades do CI-FCUL (já identificadas no capítulo anterior em "Características Pretendidas no CMS") foi naturalmente o primeiro passo dado no estudo. Apenas os CMS que na sua descrição parecem satisfazer de facto as propriedades mais importantes, foram considerados como ponto inicial de estudo.

Para além disso, comecei também por definir os tipos de linguagem que poderiam ser usadas por cada Gestor de Conteúdos. Assim, foram essencialmente considerados (com apenas duas excepções) CMS que têm como linguagem de programação as linguagens JAVA ou PHP. Isto deve-se a dois factores:

1) Como já referi anteriormente, o desenvolvimento das aplicações no CI-FCUL, é no momento deste estudo, feito em linguagem PHP, e portanto a preocupação por procurar integração imediata com esse tipo de desenvolvimento, é meio caminho andado para no futuro se conseguir o mesmo com a plataforma de desenvolvimento rápido. Uma boa plataforma gestora de conteúdos implementada em PHP será à partida completamente integrável com uma plataforma de desenvolvimento que faça a implementação de conteúdos do mesmo género, podendo essa integração ser feita, caso necessário, através de uma terceira linguagem (como é o caso da linguagem Ajax) ou através da chamada de URLs referentes cada componente implementada.

2) Uma vez que a plataforma para rápido desenvolvimento de aplicações DIF2.0 (que está a início definida para utilização no desenvolvimento de projectos, mas que será apenas descrita mais à frente neste documento), utiliza linguagem de programação JAVA, um CMS capaz de desenvolver conteúdos nessa mesma linguagem facilita de forma óbvia a posterior integração de conteúdos dinâmicos e estáticos e respectiva camada de apresentação dos mesmos. Esta integração, sendo executada na mesma linguagem, poderá à partida ser efectuada por via programática no software Eclipse utilizado pela DIF2.0 como ambiente de desenvolvimento. O facto de nesta, qualquer componente ser implementada através da importação e do uso de propriedades guardadas em bibliotecas de extensão ".Jar", possibilita pela mesma via o uso de bibliotecas de componentes exportadas directamente do CMS.

Após limitar as opções a estas duas linguagens, a procura de CMS possíveis de utilização fez-se através da consulta de diversos fóruns e sítios dedicados ao assunto (os dois sítios Web que mais contribuíram para este efeito foram referidos anteriormente), através do "feedback" dado por diversos

utilizadores das diversas ferramentas open-source, e naturalmente e como factor determinante, através da experimentação dos vários sugeridos pela pesquisa. Desta forma cheguei à seguinte lista de possíveis gestores de conteúdo para o CI-FCUL (tabela 5):

CMS:	**Linguagem:**
Alfresco	Java
Magnolia	Java
OpenCMS	Java
Concrete5	Lamp (PHP)
EZPublish	Lamp (PHP)
Umbraco	.Net
Plone	Phython
Jahia	Java
Joomla	Lamp (PHP)
Drupal	Lamp (PHP)

Tabela 5: Lista de Gestores de Conteúdos Abordados

Em anexo a este relatório (Apêndice C), faço a análise e relato pormenorizados de cada uma destas ferramentas e sua respectiva experimentação, identificando as suas principais características, vantagens e desvantagens de uso, concluindo sobre o seu uso ou não adopção no Centro de Informática da FCUL.

Como forma de revisão a todo esse estudo apresento de seguida um pequeno resumo que permite ter uma visão geral do mesmo sem que no entanto se entre em pormenores demasiado específicos de cada um.

4.1.3 Escolha do CMS:

Resumo do Estudo:

De todo o estudo e leitura atenta dos vários sistemas de gestão de conteúdo abordados neste relatório é já possível perceber as duas soluções mais adequadas para uso e adopção no CI-FCUL tendo bastado para isso perceber quais destas plataformas conseguem ao mesmo tempo implementar um maior número de funcionalidades requeridas pelo Centro de Informática e deter um menor número de fraquezas ao nível do uso e de uma possível integração com outras ferramentas de desenvolvimento (Tabela 6).

Percebendo-se a partir deste estudo que Joomla e Drupal são os dois gestores de conteúdo Web mais adequados e mais estáveis para adopção no Centro de Informática da FCUL, interessará comparar as duas ferramentas pormenorizadamente, no que a funcionalidades diz respeito, mas também tendo em conta a utilização futura da ferramenta adoptada e a forma como poderá ser integrada com outras ferramentas.

De qualquer forma, e como forma mais imediata de perceber qual é de facto o melhor sistema de gestão de conteúdos, exponho de seguida um quadro (Tabela 6) que recapitula as funcionalidades de cada sistema, as fraquezas encontradas e compara as várias plataformas ao nível da sua comunidade, simplicidade, e outros critérios relevantes.

A coluna referente à classificação é calculada com um simples somatório das várias classificações dadas nesta tabela. Atribuindo valores de 1,2 e 3 para respectivamente as cores vermelho, amarelo e verde, e seus respectivos significados, tem como único objectivo funcionar como um somatório de qualidade, não tendo em conta os graus de importância de umas funcionalidades relativamente a outras. A classificação máxima possível, correspondente a ter a cor verde em todas as atribuições seria de 54 pontos.

As funcionalidades e propriedades presentes neste quadro de classificações são apenas as que se consideram essenciais e que tem sentido serem comparadas ao nível dos vários sistemas de gestão de conteúdos estudados. Exemplo disto, é que a própria propriedade de “Comunidade: Módulos e Funcionalidades” têm precisamente como objectivo a avaliação da capacidade de extensão dos vários sistemas no que a funcionalidades diz respeito.

Através da leitura da tabela e das classificações dos diferentes gestores de conteúdos, pode-se confirmar mais uma vez a posição dos CMS Drupal e Joomla como os dois melhores e mais versáteis para adopção. Interessa agora fazer uma comparação entre estes dois para perceber qual aquele sobre o qual deve recair a escolha.

CMS	Alfresco	Magnolia	OpenCMS	Concrete5	EzPublish	Umbraco	Plone	Jahia	Joomla	Drupal
Criação de conteúdos	Difícil	Média	Fácil	Fácil	Fácil	Fácil	Média	Fácil	**Média**	**Fácil**
Diversidade Conteúdos	Elevada	Média	Média	Baixo	Elevada	Elevada	Média	Baixa	**Elevada**	**Elevada**
Versionamento	Sim	Sim	Não	Sim	Sim	Sim	Sim	Não	**Não**	**Sim**
Regras e Fluxos de Tarefas	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	**Sim**	**Sim**
Utilizadores e Privilégios	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	**Sim**	**Sim**
Calendarização de Acções:	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim	**Sim**	**Sim**
Compatibilidade Externa	Média	Média	Média	Média	Média	Alta	Alta	Baixa	**Alta**	**Alta**
Templates: Variedade de Apresentação	Desconh.	Média	Baixa	Alta	Média	Alta	Média	Média	**Alta**	**Alta**
Templates: Nível de Personalização	Desconh.	Baixa	Baixa	Baixa	Média	Média	Baixa	Baixa	**Média**	**Média**
Templates: Facilidade de Implementação	Desconh.	Média	Difícil	Média	Média	Média	Difícil	Média	**Média**	**Média**
Explorador de Ficheiros	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Não	**Sim**	**Não**
Sistema de Pesquisa de Conteúdos	Sim	Sim	Não	Não	Sim	Sim	Sim	Sim	**Sim**	**Sim**
Comunidade: Resolução de Problemas	Pobre	Pobre	Médio	Pobre	Médio	Médio	Rico	Pobre	**Rico**	**Rico**
Comunidade: Módulos de Funcionalidades	Médio	Pobre	Médio	Médio	Médio	Médio	Rico	Pobre	**Rico**	**Rico**
Comunidade: Documentação	Pobre	Médio	Rico	Pobre	Médio	Rico	Rico	Pobre	**Rico**	**Rico**
Simplicidade na Administração (1 a 3)	Média	Média	Médio	Simples	Médio	Simples	Simples	Simples	**Simples**	**Simples**
Simplicidade na Utilização (1 a 3)	Média	Média	Médio	Simples	Sim	Médio	Médio	Simples	**Simples**	**Simples**
Facilidade de Aprendizagem (1 a 3)	Difícil	Média	Médio	Simples	Sim	Médio	Médio	Simples	**Fácil**	**Fácil**
Somatório Geral:	**36 (10º)**	**39 (6º)**	**37 (9º)**	**39 (6º)**	**44 (5º)**	**48 (3º)**	**46 (4º)**	**39 (6º)**	**49 (2º)**	**50 (1º)**

Tabela 6: Tabela de classificação dos CMS estudados

DRUPAL como solução:

Quando comparamos DRUPAL e JOOMLA à partida, e olhando tanto para o estudo mais elaborado como para a tabela de classificações do capítulo anterior, não nos parece haver uma grande diferença, ou

uma clara superioridade de uma das plataformas em relação à outra. No entanto, após uma experimentação mais aprofundada de ambos e uma integração mais convicta e real no seio das duas comunidades de utilizadores, acabamos por compreender uma superioridade do Drupal em relação ao Joomla.

Indo por partes, identifico e comparo os dois relativamente às suas principais diferenças:

Interface:

No Drupal toda e qualquer criação de conteúdo faz-se numa típica navegação pelo próprio sítio Web em construção, não existindo uma delimitação real entre este e a interface de administração onde se encontram acessíveis todos as configurações, menus de instalações, etc. Para isto, o Drupal conta apenas com um menu de administração situado no sítio em construção, através do qual se acede a toda e qualquer operação.

Pelo contrário, no Joomla existe uma separação entre o próprio sítio Web em construção onde se pode efectuar operações de edição sobre os conteúdos e uma interface de administração onde se pode aceder a todas as outras propriedades e funcionalidades através de um típico "painel de controlo" (Figura 8).

Figura 8: Menu de administração do Drupal e "painel de controlo" de administração do Joomla

Qualquer das alternativas é eficaz à sua maneira, e embora no Drupal haja uma primeira impressão de estranheza de utilização, após um período muito curto de aprendizagem, a sua utilização revela-se na verdade mais eficaz que a do Joomla. A possibilidade de ter sempre acesso a qualquer funcionalidade ou configuração através de um simples menu, torna todo o processo de criação "mais próximo". Para além

disso, o facto de se estar sempre inserido no sítio Web em construção ajuda a ter uma real ideia do funcionamento deste à medida que se vai formulando como um serviço de Internet.

Módulos:

O que torna o Drupal superior ao Joomla, no entanto, acaba por não ser só a sua utilização e a forma mecanizada como o Drupal vai crescendo com a utilização. Sabendo que em termos de comunidades de utilizadores e suporte os dois se equivalem, é importante perceber a extensão dos módulos produzidos por cada uma destas comunidades.

Em primeiro lugar, verifica-se que no Drupal a instalação de módulos é feita de uma forma muito mais simples que no Joomla. No primeiro, o descarregamento de um módulo e colocação deste numa pasta própria de módulos é suficiente para que "apareça" na lista de módulos instaláveis e acessíveis com um clique de rato. No segundo, após descarregamento do módulo do sítio que o fornece, é necessário o seu carregamento para a ferramenta, a sua activação em pelo menos dois menus e ainda a sua configuração elaborada para que o seu funcionamento seja o correcto. Neste aspecto o Drupal é novamente mais simples de utilizar, e assim que instala um par de módulos o utilizador já está familiarizado com o processo.

Em segundo lugar, a quantidade de módulos disponibilizados por ambas as comunidades é imenso No entanto, a procura de módulos, a diversidade destes e a forma como qualquer utilizador pode iniciar o desenvolvimento de um módulo seu, volta a superiorizar o Drupal em relação ao Joomla. O sítio oficial do Drupal é também o seu principal repositório de extensões, tal como é o seu principal fórum e centro de utilizadores, e qualquer funcionalidade desejada é normalmente encontrada, explicada e se necessário é dado o devido suporte por parte de uma comunidade "desejosa" por ajudar. Já no Joomla, existe um grande número de extensões no sítio oficial, tal como existe um grande número destas fora do sítio oficial, no entanto a forma como um utilizador pesquisa pelo módulo desejado e frequentemente encontra ligações para conteúdos pagos produzidos por outras empresas, e a forma como o suporte é dado fora do contexto dessa mesma funcionalidade (num fórum à parte) jogam a seu desfavor.

Em terceiro lugar, após alguma utilização das duas ferramentas, fica a clara ideia de que no Drupal é possível fazer praticamente qualquer coisa que se queira, existindo um módulo para quase tudo numa simplificação extrema de implementação de funcionalidades que normalmente demoram bastante tempo a programar num sítio Web. Para além disto, a principal falha ao nível de extensões de funcionalidades no Joomla relaciona-se com a segurança do sistema, permissões de utilizadores e optimizações para motores de busca. Uma vez que a componente de segurança aqui latente é uma das mais importantes ao nível de um CMS, é evidente a superioridade do Drupal.

Futuro:

Percebendo-se a superioridade do Drupal tanto na utilização como na modularidade e na quantidade de funcionalidades que permite implementar, interessa por fim referir a sua superioridade em termos de futuro e em termos de integração com outras plataformas. Existe no presente um grande número de

utilizadores a adoptar o Drupal como sua ferramenta de gestão de conteúdos o que causa um crescimento ainda maior na comunidade. No movimento inverso está o Joomla, havendo inclusive um grande número de utilizadores a fazerem a transição deste para o Drupal.

Por fim, considerando a temática já referida da integração do CMS adoptado com uma ferramenta de desenvolvimento para sítios Web com grande número de conteúdos dinâmicos, e possibilidade de integração de uma outra plataforma CMS é praticamente igual, no sentido em que a utilização de conteúdos gerados se fará com o auxílio a técnicas PHP e Ajax e neste capítulo, tanto Drupal como Joomla são idênticos no código fonte das páginas de conteúdos que geram.

Conclusão do Estudo:

Por todo o estudo feito, pelas ilações que foram sendo retiradas em todo este relatório, a adopção do CMS Drupal como ferramenta de gestão de conteúdos Web do Centro de Informática da FCUL é de forma evidente a melhor solução para o futuro do Centro e permite aguardar com grande expectativa o início de criação de serviços com o auxílio a esta poderosa ferramenta de gestão e criação de Sítios Web.

4.2 Adopção de uma Plataforma Rápida de Desenvolvimento

Estando definida uma metodologia de desenvolvimento a implementar no CI-FCUL, é necessário naturalmente proceder à adopção de uma ferramenta que permita um rápido desenvolvimento de aplicações e que possibilite ao mesmo tempo a integração de informação criada através de ferramentas como Gestores de Conteúdos (abordado no capítulo anterior). No caso da FCUL, o desenvolvimento existente baseia-se apenas em pura codificação em linguagem PHP de inteiros serviços Web, sem que se faça qualquer uso de ferramentas automáticas ou semi-automáticas de geração de funcionalidades e seu código respectivo. A adopção de uma plataforma de rápido desenvolvimento para a implementação de aplicações e serviços para a FCUL foi desde o início deste projecto um principal ponto de foco e interesse. No entanto e ao contrário do outro tipo de ferramentas usadas para auxiliar o desenvolvimento (Gestores de Conteúdos), não foi efectuado qualquer estudo analisador do melhor produto open-source a utilizar no que respeita a este tipo de plataformas, tendo sido desde início adiantada a ideia de adopção de uma plataforma desenvolvida por uma empresa já familiarizada com a FCUL – a Digitalis – e que dá pelo nome de DIF2.0.

Esta é a plataforma de desenvolvimento que passo a descrever em primeiro lugar neste capítulo, apresentando depois algumas plataformas alternativas para o mesmo fim de desenvolvimento, para por fim discutir a adopção da plataforma da Digitalis apresentando as razões que sustentaram essa adopção ao mesmo tempo que discuto em que medida as plataformas alternativas estudadas a poderiam superar.

4.2.1 DIF2.0

A segunda versão da Digitalis Internal Framework, referida como DIF2.0, é uma plataforma de aplicações em Java que tem por objectivo conferir facilidade, flexibilidade e simplicidade no desenvolvimento de aplicações e serviços Web, sendo classificada como uma plataforma RAD (Rapid Application Development) ou seja como uma plataforma para rápido desenvolvimento de aplicações, servindo-se e impondo convenções de configuração e codificação, bem como fazendo uso de bibliotecas de funções ou funcionalidades já implementadas como forma de acelerar o desenvolvimento de aplicações e serviços ***[Digitalis, 2008]***.

Principais Características:

A forma mais simples de definir esta plataforma de desenvolvimento é através da enumeração dos principais pontos que a caracterizam como RAD ***[Digitalis, 2008]***:

- Possui mecanismos que aumentam a velocidade de execução e o desempenho no desenvolvimento.
- Oferece suporte e integração de funcionalidades Web2.0.

- É uma plataforma extensível que permite aos desenvolvedores a implementação de novas convenções sobre código ou funcionalidades.
- Não possui estruturas XML de difícil entendimento e apresenta por isso bastante facilidade de configuração.
- Baseia-se na geração automática de código Java a partir de simples anotações.
- Baseia-se na ideia de "preenchimento do espaço em branco" (preenchimento de métodos em que o código é parcialmente gerado a partir das anotações) por parte do programador.
- Assenta numa ferramenta de gestão da plataforma – Maven – que permite configurar vários aspectos de funcionamento desta.
- Oferece suporte a acessibilidade e a diferentes linguagens.

Estas características traduzem-se num conjunto de pontos marcantes que ajudam a perceber a DIF2.0 na sua globalidade:

Integração e Flexibilidade na DIF2.0

A DIF2.0 traduz-se por um conjunto de módulos que podem ser "montados" de forma diferenciada para possibilitar a adaptação a diferentes usos. Assim, nesta plataforma, o programador pode costumizar a seu gosto bem como criar os seus módulos., e por módulos entende-se desde componentes de arquitectura a funcionalidades de ponta como as que estão relacionadas com o uso de bases de dados como fontes de informação para uma aplicação, ou funcionalidades relacionadas com autentificação, autorização, segurança, etc. Todas as capacidades de flexibilidade oferecidas pela plataforma relacionam-se com um mecanismo de inversão de controlo (IoC), em que a sequência de chamada dos métodos não é directamente determinada pelo programador mas delega numa infra-estrutura denominada de "container" que fica com controlo sobre a execução da aplicação ***[Digitalis, 2008]***.

Desempenho na DIF2.0

Tal como o uso de geração automática de código permite aumentar o tempo de execução e traduz-se numa maior eficiência na implementação de novas aplicações, também o uso de caches para dados significativos e para um carregamento mais eficiente de funções usadas em maior número de vezes, confere maior desempenho a esta ferramenta de desenvolvimento. Mas acima de tudo, o facto de a DIF2.0 possuir um módulo arquitectural que permite adaptar o sistema a diferentes requisitos, alterando os níveis de utilização de memória, tem importância central no desempenho de toda a ferramenta ***[Digitalis, 2008]***.

Componentes da DIF2.0

Tendo uma arquitectura baseada em módulos (Figura 9), apresento de seguida uma breve descrição dos módulos que fazem parte da arquitectura da plataforma de desenvolvimento da Digitalis ***[Digitalis, 2008]***:

- Núcleo (Core) – Constituído por diversas camadas em que cada uma oferece um serviço aos clientes da plataforma:
    - **ChAL**: Camada do canal de abstracção que fornece independência nas suas tecnologias de transmissão e apresentação.
    - **"Dispatcher"**: Contém a lógica de execução que corre em cada pedido de serviço e é responsável por passos de validação, autentificação, autorização, execução e conclusão dos mesmos.
    - **DEM – Meta modelo de Entidades DIF:** Define a hierarquização estrutural dentro da plataforma, essencial para a sua gestão, através de um conjunto de elementos de código.
    - **Geração Dinâmica de Código:** Camada responsável pela geração automática de código que se baseia em anotações Java, oferecendo-as num lote relativamente grande de forma a cobrir um grande número de funcionalidades. Esta camada permite retirar carga programática ao utilizador comum da plataforma e possibilita a extensão constante da mesma.
    - **IoC – Inversão de Controlo**: Camada que possibilita a integração com outras tecnologias, seguindo o princípio de separação de conceitos.
    - **Segurança:** Camada que possibilita a configuração de diferentes níveis de segurança de acordo com cada aplicação fazendo uso de gestores de identidades, de autentificação e de autorização.
    - **Gestor de Parâmetros:** Camada que permite à plataforma validar parâmetros automaticamente, convertendo os vários tipos para Java.
    - **Mensagens:** Camada que permite costumização de mensagens e facilita a tradução de aplicações.
    - **"Error Reporting":** Camada encarregada de facilitar e tornar acessível a detecção de erros de programação.
    - **Configuração:** Camada de configuração que permite ao utilizador abstrair-se de pormenores demasiado técnicos.
    - **Biblioteca de Componentes Web 2.0:** Camada que disponibiliza a utilização de componentes programáticas e sua integração no desenvolvimento de aplicações. Existe uma larga biblioteca de componentes disponível (e em crescimento), incluindo integração com JavaScript e Ajax.

- **Maven:** Camada de configuração que suporta grande parte da facilidade de uso da plataforma.

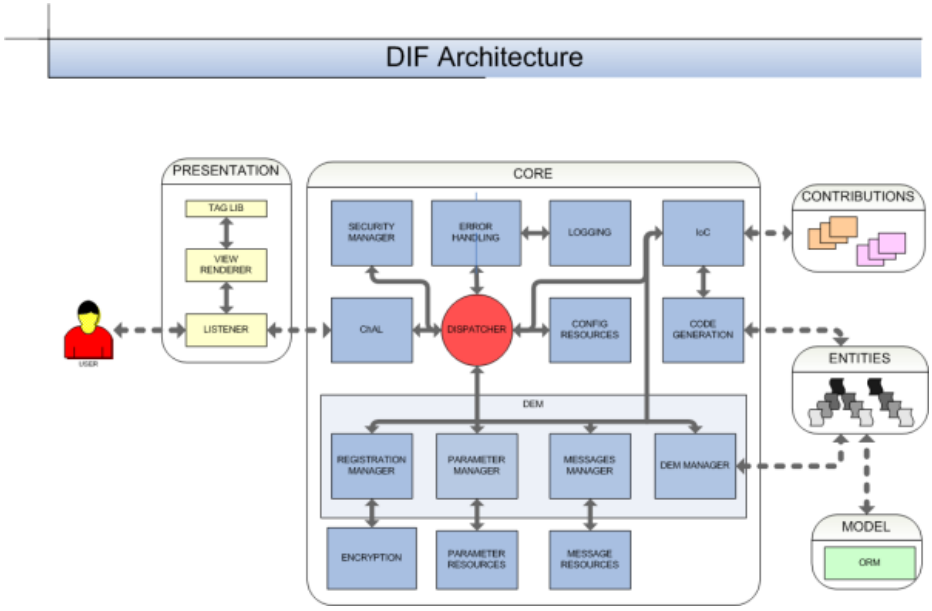


Figura 9: Arquitectura e respectivas camadas da Plataforma DIF2.0 (Imagem retirada do Sítio Oficial da Digitalis)

Para além do núcleo propriamente dito, e como é facilmente identificável na imagem acima, existe ainda um constituinte relacionado com a integração com bases de dados relacionais – Modelo ORM – que procura o mapeamento de informação contida em repositórios externos de informação, e fazendo uso das capacidades de geração automática de código, cria os métodos necessários ao tratamento desses dados por parte da plataforma ou das aplicações.

Esta descrição procurou apenas resumir de uma forma relativamente simples todos os pormenores da arquitectura desta plataforma de desenvolvimento. Para acesso a uma descrição mais aprofundada de cada um dos constituintes recomenda-se a leitura atenta da documentação presente no sítio da plataforma em causa, acessível em http://netpa.digitalis.pt/apache2-default/dif2/index.html.

Funcionamento Geral da Plataforma

A descrição da plataforma de desenvolvimento só por si não é suficiente para a percepção das suas reais capacidades, sendo necessário proceder a uma descrição mais elaborada do funcionamento geral da ferramenta. Para uma eficaz descrição, irei descrever a plataforma em termos de Funcionamento Base, e em termos de Pormenores Técnicos:

Funcionamento Base:

Baseado na plataforma de desenvolvimento Eclipse IDE (www.eclipse.org) (Figura 10) e correndo sobre Maven/Apache (http://maven.apache.org/), a DIF2.0 é como já disse, um conjunto de módulos de funcionalidades, em que cada módulo é constituído por um conjunto de classes e funções escritas em Java. Por isso, o funcionamento básico de um desenvolvedor servindo-se desta plataforma é a

programação de conteúdos com a utilização de métodos contidos em bibliotecas de classes Java e que permitem simplificar a implementação das funcionalidades mais comuns, reduzindo-as a uma série de linhas, ou à simples utilização de comentários que desencadeiam a geração automática de código das respectivas funcionalidades.

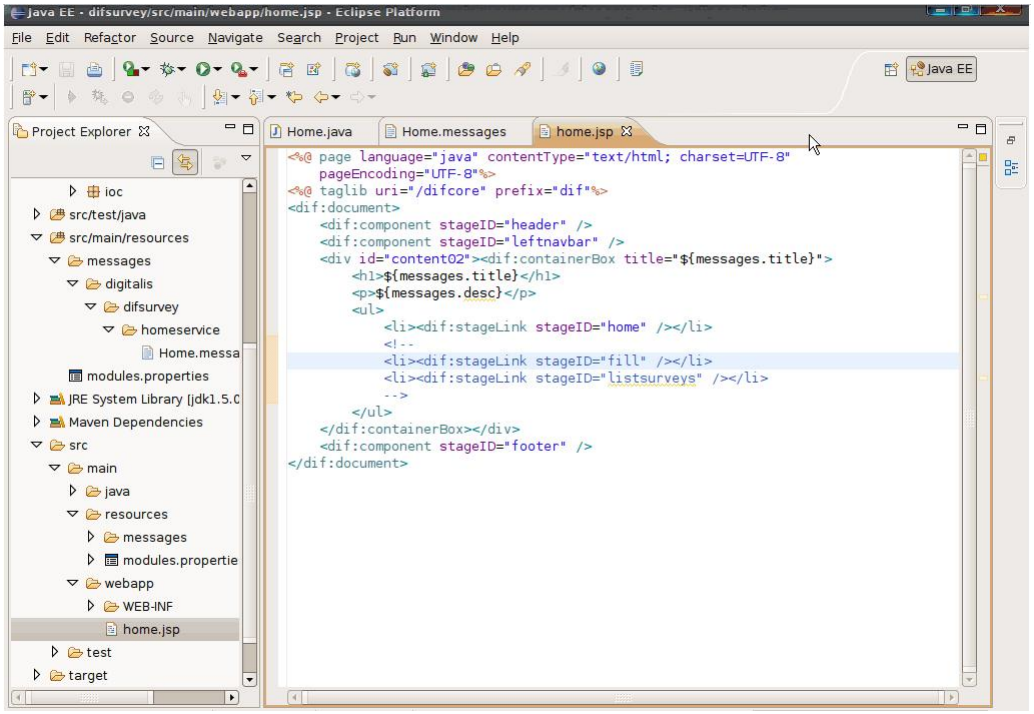


Figura 10: Imagem da interface do Eclipse em funcionamento com DIF2.0

A típica criação de um projecto utilizando a DIF2.0, é assim, normalmente caracterizado pela seguinte sequência de acções:

1. Escrita de uma linha de comando em Maven para criação do projecto.

2. Escrita de nova linha de comando em Maven para criação da estrutura do projecto no Eclipse. Este comando desencadeia por si só a criação das pastas e ficheiros necessários a um novo projecto, sem necessitar de qualquer intervenção do desenvolvedor.

3. Uso da plataforma (Eclipse com bibliotecas DIF2.0) para codificação de funcionalidades e da aplicação.

4. Compilação da aplicação em desenvolvimento fazendo uso de comandos Maven, que por si só desencadeiam o lançamento da aplicação no navegador Web.

5. Visualização e experimentação da aplicação em desenvolvimento.

A implementação de funcionalidades fazendo uso dos típicos comentários processa-se com a inclusão de pedaços de código precedidos do sinal “@” como por exemplo (Figura 11):

```
@ServiceDefinition(name = "Fill Survey Service", application="difsurvey")
public class FillService {

}
```

Figura 11: Exemplo de anotação para geração automático de código na DIF2.0

Estas linhas de comentário substituem muitas mais que serviriam para a implementação das funcionalidades a que dizem respeito (no caso acima mostrado, a anotação especifica o tipo de entidade que a classe representará). Para uma melhor ideia do funcionamento deste tipo de comentários, recomenda-se a leitura da documentação online no site oficial da DIF2.0 em http://netpa.digitalis.pt/apache2-default/dif2/documentation.html.

Tal como através da utilização de comentários, é ainda prática corrente, que componentes que necessitam de grande esforço de desenvolvimento estejam disponíveis de forma "out of the box", sendo possível a sua utilização simplificada em qualquer aplicação em desenvolvimento. Essa utilização baseia-se na típica invocação de métodos na programação de funcionalidades, métodos esses, importados de bibliotecas Java. Exemplo típico de uma componente destas, é a de ajuda ao preenchimento de formulários online, com correcção ou alertas automáticas no preenchimento de campos.

Pormenores Técnicos:

Após percepção do funcionamento básico da plataforma de desenvolvimento em causa, interessa destacar um conjunto de pormenores técnicos essenciais para um melhor entendimento da mesma e suas capacidades e limitações. Estes pormenores técnicos vão desde instruções de utilização, a guias de programação a pormenores relacionados com certas componentes ou camadas da DIF2.0 ***[Digitalis, 2008]***:

- **Hierarquização estrutural na DIF2.0:** Na criação de aplicações nesta plataforma existe uma estruturação típica da informação e uma hierarquia clara em termos de serviços, aplicações e funcionalidades (já descrito acima como DEM). Assim temos a seguinte hierarquia na construção de aplicações:
    - **"Provider"** – Em português "Fornecedor", será sempre no caso de desenvolvimento de novas aplicações o Centro de Informática ou a Digitalis, no caso de se fazer uso de modelos já definidos.
    - **Aplicação** – Por cada aplicação construída haverá uma entidade deste tipo que a refere no volume de código da DIF2.0. Para cada "Provider" existirão várias aplicações.
    - **Serviço** – Podem existir vários serviços por Aplicação, e cada serviço reflecte a implementação de uma funcionalidade geral ou conjunto de funcionalidades que constituem um serviço para o utilizador da aplicação.
    - **Etapa** - Representa (e implementa) tipicamente uma funcionalidade na aplicação. Cada serviço é constituído tipicamente por uma ou mais Etapas. É a este nível que se apresenta a maior parte do código implementado na DIF2.0.

A definição de cada elemento dos acima indicados é relativamente simples, bastando proceder à anotação de uma classe Java para que possam ser acedidos programaticamente via API da plataforma.

- **Uso e Adição de Anotações na DIF2.0:** No que diz respeito à utilização de anotações, é possível a qualquer desenvolvedor a adição de novas anotações à plataforma para inclusão simplificada de funcionalidades já desenvolvidas. Essa inclusão na plataforma faz-se através de dois simples passos:
    - Criação da anotação, através de texto normal que não necessita de qualquer compilação.
    - Definição da lógica associada à geração de código, através de código Java que vai ser compilado internamente.
- **Adição de Módulos de Funcionalidades:** Uma forma mais fácil de propiciar a utilização de funcionalidades que agregando lógicas de programação a anotações, é simplesmente a adição de módulos de funcionalidades à biblioteca da plataforma de desenvolvimento. Para isto não é necessária qualquer reconfiguração ou recodificação, mas apenas os dois seguintes passos:
    - Funcionalidades são "empacotadas" num ficheiro JAR com um descritor do ficheiro que declara os nomes das classes que fazem a ligação entre interfaces e implementações.
    - Ficheiro JAR é carregado num servidor da aplicação e lido pela camada de IoC.
- **Ficheiros de Mensagens:** Para possibilitar a transcrição de qualquer aplicação para diferentes línguas, a DIF2.0 baseia as suas mensagens em ficheiros de propriedades que fazem corresponder linhas de texto aos diferentes eventos de impressão de mensagens nas aplicações. A cada ficheiro de mensagens é dado o mesmo nome que à entidade a que se refere, seguido do sufixo ".messages". Para além disto:
    - Anexar uma extensão de linguagem ao nome do ficheiro de mensagens, permite a sua utilização em várias línguas.
    - As mensagens costumizadas por utilizadores são guardadas num repositório externo.

Após a descrição da ferramenta de desenvolvimento da Digitalis e após a percepção do seu funcionamento, interessa analisar da mesma forma plataformas alternativas que permitam atingir o mesmo fim no que diz respeito ao desenvolvimento de aplicações pelo Centro de Informática. Só após esta análise se poderá concluir sobre o uso da plataforma adoptada a princípio, e perceber se terá sido tomada a decisão mais acertada ou se por outro lado o CI-FCUL ganharia mais com a adopção de outra plataforma não contemplada a início.

4.2.2 Breve Análise a Alternativas à DIF2.0

Não tendo sido feito qualquer estudo prévio relacionado com alternativas à Digitalis Internal Framework, são adiantadas de seguida, e de forma relativamente breve, algumas plataformas capazes de levar a cabo o mesmo tipo de desenvolvimento de aplicações, e que noutras circunstâncias, teriam sido consideradas

para adopção no Centro de Informática da FCUL. Em termos de análise, interessa referir que para cada ferramenta a seguir especificada, irá ser fornecida uma descrição com as características que se destacam, uma ideia do seu funcionamento, e um contrabalanço entre vantagens e desvantagens na sua adopção. São abordadas duas plataformas: CakePHP e Outsystems Platform.

CakePHP:

O CakePHP é uma plataforma rápida de desenvolvimento para PHP que fornece uma extensa arquitectura para desenvolvimento, manutenção e produção de aplicações. Fazendo uso de padrões de desenho bem conhecidos como MVC (separação entre a lógica de negócio e lógica de apresentação, por forma a permitir o desenvolvimento, teste e manutenção de forma isolada) e ORM (uso de bancos de dados relacionais no desenvolvimento), esta plataforma reduz os custos do desenvolvimento e ajuda o desenvolvedor a escrever menos código para implementação das mesmas funcionalidades ***[Cake, 2010]***.

Características da Plataforma

Uma melhor forma de perceber as capacidades da plataforma é enumerando e explicando as características que fazem dela uma plataforma de desenvolvimento rápido de aplicações ***[Cake, 2010]***:

- **Sem Configuração:** Basta configurar a base de dados e a plataforma automaticamente procede às configurações necessárias.

- **Extremamente Simples:** Todo o funcionamento com a plataforma não difere muito da típica programação em PHP, sendo que no entanto a implementação é tornada mais simples, reduzindo o esforço e o número de linhas necessárias de codificar para implementação de funcionalidades.

- **Comunidade Activa:** Vasto uso por todo o mundo Web resulta na existência de muita documentação relativa à plataforma, bem como em fóruns bastante activos e na fácil resolução de problemas e dúvidas.

- **Código Limpo:** Qualquer linha de código de funcionalidades utilizáveis, foi escrito e programado pela equipa de desenvolvimento da CakePHP e por isso obedece aos mesmos padrões de programação.

- **Implementa as Melhores Práticas:** Funcionalidades implementadas de origem na plataforma, e utilizáveis em qualquer projecto a desenvolver, cobrem aspectos de segurança, autentificação, gestão de sessões e muitas outras.

- **Aprendizagem Amigável:** Tanto para programadores experientes como para iniciantes, a aprendizagem da plataforma é amigável e relativamente fácil, existindo desde o primeiro momento uma sensação de conforto na aprendizagem e gratificação na experimentação.

- **Geração de Código:** A utilização de funções já implementadas de origem, e que visam facilitar toda a implementação de funcionalidades típicas de uma aplicação Web, resulta na geração do código PHP respectivo, sem que haja qualquer intervenção ou instrução do desenvolvedor nesse sentido. Essa geração de código está completamente isenta de erros.

- **Validação de Modelos de Dados:** A validação de dados em toda a implementação é relativamente simples e assegura a persistência das aplicações construídas, bem como, simplifica mecanismos de inserção de dados.

- **Arquitectura Simples**: Arquitectura baseada em vistas e controladores, em que é definido uma espécie de mapa geral da aplicação no controlador, e para cada pedido é atribuído uma vista que normalmente apresenta informação contida em modelos ou em bancos de dados.

- **Ajudas de Programação:** Toda a programação é suportada por ajudas de implementação, como componentes HTML, facilitadores de criação de formulários, paginação, uso de AJAX, uso de Javascript, RSS, etc.

Funcionamento da Plataforma

De seguida é dada uma ideia do funcionamento do CakePHP, para que seja possível entender a simplicidade de processos de que se falou até agora na descrição da plataforma.

No fundo, o funcionamento base desta plataforma apoia-se no uso de funções PHP que implementam funcionalidades tipicamente presentes em serviços e sítios Web, e que são disponibilizadas no desenvolvimento de qualquer aplicação, como forma de facilitação de toda a implementação de serviços. Para isso a plataforma necessita, tal como quase todas as plataformas de desenvolvimento, de uma ferramenta de implementação intermediária, que no caso é o Eclipse (www.eclipse.org), onde os desenvolvedores possam escrever todo o código das aplicações (Figura 12).

Em CakePHP existe tipicamente um controlador de páginas que funciona como um mapa geral de toda a aplicação e que tem a função de especificar cada entrada na aplicação através das moradas URL pretendidas. Assim, para cada função no controlador de páginas, existe uma vista desenvolvida num ficheiro extensão ".ctp", que implementa uma qualquer funcionalidade, e que pode ir desde a apresentação simples de texto, à codificação de uma inteira funcionalidade, à apresentação de dados presentes numa base de dados, etc., e que representa a página a ser carregada pelo navegador de cada vez que é inserido o nome da função especificada na morada URL.

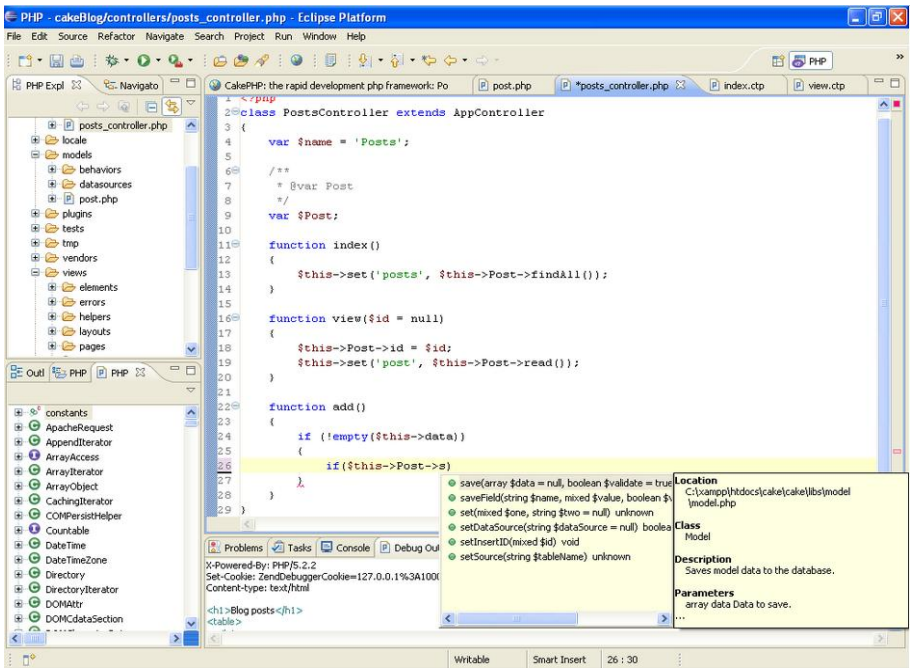


Figura 12: Imagem do controlador com páginas da aplicação definidas sobre funções

Outra das características principais desta plataforma, é como já disse, a simplicidade no uso de bases de dados como fonte da informação a apresentar nas aplicações desenvolvidas. Retornar instâncias de uma tabela uma por uma ou retornar toda a informação contida numa tabela, tem praticamente o mesmo grau de dificuldade, sendo esse grau de dificuldade muito baixo, e baseando-se na chamada a métodos, como a seguir demonstrado:

```
function view($id = null)
{
    $this->Post->id = $id;
    $this->set('post', $this->Post->read());
}
```

Figura 13: Pedaço de código que demonstra retorno de informação de uma base de dados

Como se pode verificar (Figura 13), o método “view” especificado no controlador, recebe um argumento referente ao id de um “post”, e através das duas linhas seguintes é devolvido o objecto que apresenta esse id ao nível da base de dados e mostrado ao nível da aplicação. A forma como este é apresentado, é definida num ficheiro obrigatoriamente denominado “view.ctp”, onde são especificados os campos a mostrar na aplicação.

Da mesma forma que se processa a chamada a estes métodos, todas as outras funcionalidades “empacotadas” no CakePHP e que facilitam a implementação de aplicações, são utilizadas da mesma forma, não exigindo inclusive grande aprendizagem para a sua utilização. Um exemplo típico de utilização na implementação é a de modelos de criação de formulários, em que os campos de um formulário são criados de acordo com os formatos definidos na base de dados com que são relacionados, sem que o utilizador tenha que especificar mais que uma par de instruções de chamada a funções (para

além disto, é possível ainda definir regras de validação para os campos do formulário num ficheiro de configuração).

Para haver ainda uma percepção de resultados possíveis de obter com a utilização da ferramenta, mostram-se de seguida dois exemplos de produtos finais desenvolvidos com utilização de CakePHP (Figura 14):

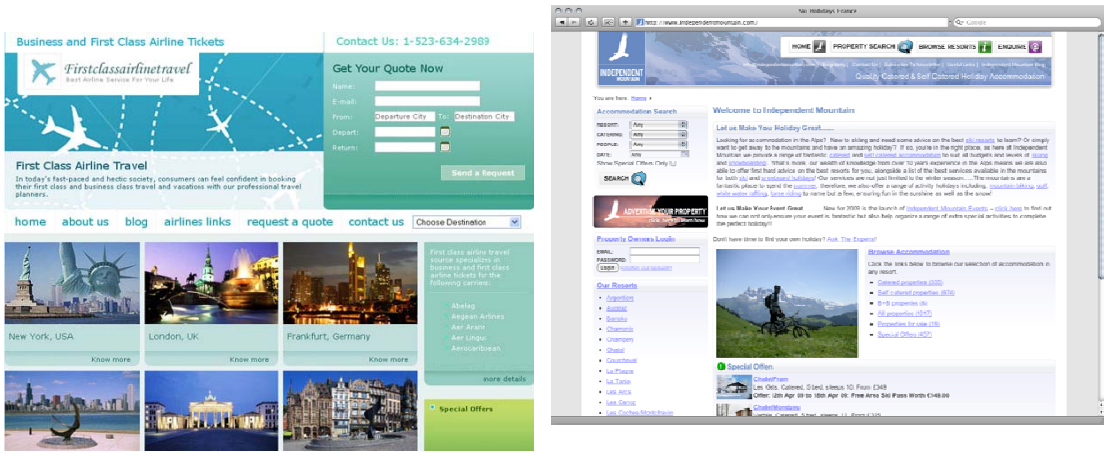


Figura 14: Aspecto de dois sítios Web desenvolvidos em CakePHP

Por fim interessa referir, que no que respeita a aspectos gráficos e de aparência de uma aplicação desenvolvida nesta plataforma, a configuração de templates é idêntica à que acontece na DIF2.0 (e em quase todas as plataformas deste tipo), tendo que haver conhecimento aprofundado de linguagem HTML e CSS, e pura codificação de propriedades visuais.

Assim sendo, e após a percepção destes exemplos de utilização da plataforma, verifica-se a sua simplicidade e facilidade de uso, e evidencia-se no capítulo seguinte, que a sua utilização poderia ser uma alternativa à plataforma adoptada.

OutSystems Agile Platform:

Traduzida para português como “Plataforma Ágil da Outsystems”, é uma plataforma de desenvolvimento que permite criar ou modificar aplicações Web usando métodos ágeis (sendo por isso tolerante a mudanças) e permitindo que o desenvolvimento seja concluído a tempo e dentro dos custos pensados, garantindo que o produto é inteiramente integrado com os outros sistemas existentes no mesmo ambiente. A plataforma de desenvolvimento da Outsystems fornece assim todas as ferramentas e funcionalidades necessárias para suportar e automatizar o desenvolvimento e gestão de aplicações de negócio, permitindo que com uma única plataforma seja possível acelerar o desenvolvimento através de: uma rápida criação de componentes conectados com sistemas e bases de dados existentes e transformação destes em blocos reutilizáveis em aplicações; definição visual de todos os aspectos das aplicações sem qualquer programação de baixo nível, incluindo definição de interfaces de utilizador, de modelos de dados, lógicas de negócio, regras de segurança e serviços Web; e tradução imediata de programação de

alto nível em produtos . Net ou Java e possibilidade de desfazer qualquer alteração e recuperar qualquer versão, de uma aplicação através dos seus mecanismos de gestão e monitorização ***[OutSystems, 2008]***.

Características da Plataforma

Para uma melhor percepção de tudo aquilo que a plataforma tem para oferecer é essencial enumerar e explicar as características que fazem dela uma plataforma ágil de desenvolvimento de aplicações ***[OutSystems, 2008]***:

- **Produção Visual de Aplicações:** Desenvolvimento de aplicações é feito de forma visual sem programação de baixo nível e forma praticamente automática. Interfaces de utilizador, modelos e bases de dados, regras de negócio e validação, serviços Web e adaptadores para integração com outros sistemas, são criados e modificados com base em arrastamento e clique numa interface visual de desenvolvimento, garantindo-se ao mesmo tempo consistência na aplicação.

- **Aplicações ricas em Funcionalidades Web:** A plataforma de desenvolvimento suporta AJAX de forma a possibilitar o desenvolvimento e integração de componentes Web 2.0 da mesma forma que são desenvolvidas componentes básicas (interface visual de desenvolvimento).

- **Suporte para Várias Linguagens:** Aplicações podem ser produzidas em diferentes língua sem qualquer necessidade de configurações ou definições complexas, contendo recursos que permitem que a tradução entre linguagens seja feita de forma automática.

- **Gestão, Controlo Centralizado:** A partir de um único ponto é possibilitado o controlo, configuração e gestão das aplicações desenvolvidas, incluindo versionamento, segurança e performance.

- **Escalabilidade e Robustez:** Aplicações são implementadas tendo em conta alta robustez e escalabilidade com possibilidade de reutilização de componentes alterando-as de forma muito ligeira.

- **Arquitectura Orientada a Serviços (SOA):** A possibilidade de "montar" serviços Web com base em ferramentas simples e intuitivas e desprovidas de programação de baixo nível, permite que toda a organização que usa a plataforma esteja orientada ao desenvolvimento de serviços.

- **Tecnologia Não-Proprietária:** A plataforma ágil da Outsystems consegue por si só gerar conteúdos . Net e Java sem ter qualquer necessidade ou dependência de interpretadores externos para essas tecnologias, tal como é capaz de criar bases de dados sem qualquer dependência de outras ferramentas.

- **Publicação e Versionamento Simplificado:** Novas aplicações e versões são automaticamente validadas e produzidas sem qualquer latência de tempo, sendo isto feito simplesmente através de

um simples clique. Também o retorno a versões anteriores é feito com base numa simples selecção.

- **Tecnologia de Mudança Embebida:** Permite que utilizadores interajam com aplicações desenvolvidas (ou em desenvolvimento) e registem directamente nestas as principais fraquezas e erros, ou simplesmente elaborem sugestões para desenvolvimento de futuras versões.

Funcionamento da Plataforma

De seguida é dada uma ideia do funcionamento da plataforma de desenvolvimento da OutSystems, para que seja possível entender a simplicidade de processos de que se falou até agora na descrição da plataforma.

O módulo de desenvolvimento da plataforma, denominado por "Service Studio", divide-se em três funções e interfaces principais: Modelação de Dados, Modelação de Processos e Interface de Desenho ***[James, 2010]***.

1. **Uso de Modelação de Dados:** O uso da função de modelação de dados é semelhante ao uso de um programa típico de criação de bases de dados tanto no que diz respeito ao tipo de funções como no que diz respeito ao aspecto da interface. Na maior parte das operações é apenas necessário a simples adição de entidades e de atributos de entidades numa pequena interface em árvore criada para o propósito. No entanto, quando é necessária uma maior configuração de atributos (como a adição de índices a entidades) está acessível uma completa interface de modelação desses dados, na qual se destaca a facilidade de criação de chaves estrangeiras entre entidades com base na definição simples do tipo de campos (e suas referências). Toda a modelação de entidades é feita assim de forma relativamente simples e uma imagem desta interface é mostrada de seguida (Figura 15).

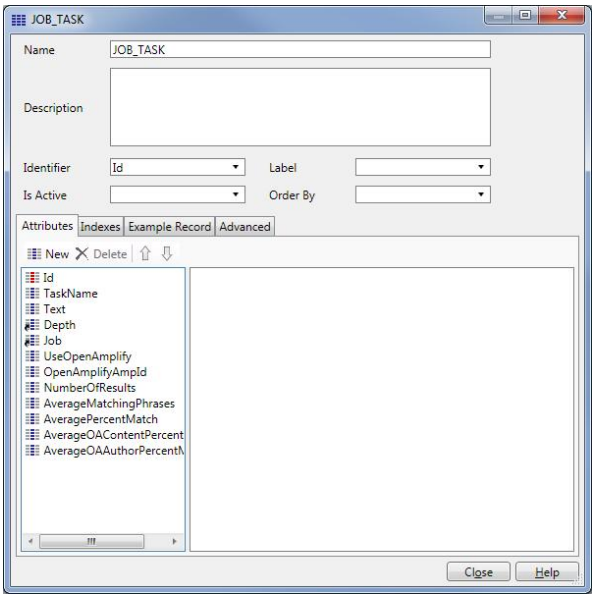


Figura 15: Ferramenta de modelação de dados no Service Studio da Plataforma da OutSystems

2. **Uso de Modelação de Processos:** Como forma de comparação, apenas se poderá dizer que esta ferramenta visual de modelação de processos apresenta semelhanças com o Microsoft Visio. No fundo o seu funcionamento baseia-se na disposição de acções ou sequências de eventos definidos por vários tipos de objectos (Figura 16). Quer o objectivo seja a definição de um formulário e respectiva interrogação de execução sobre a base de dados para preenchimento dos campos, quer se trate da actualização de uma variável, a ferramenta define a lógica das acções e as relações entre os objectos envolvidos. Cada fluxo construído irá definir uma acção que poderá depois ser reutilizada como um todo ou parcialmente como parte do fluxo de outra funcionalidade.

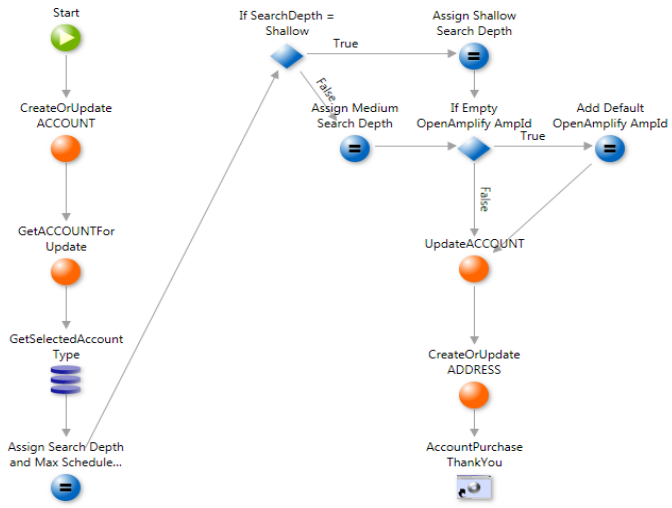


Figura 16: Exemplo de construção de um fluxo representativo de uma funcionalidade na Modelação de Processos

3. **Interface de Desenho:** A terceira interface a destacar e possivelmente a mais importante ao nível dos resultados que permite obter, é a interface de desenho baseada em ferramentas de WYSIWYG ("what-you-see-is-what-you-get") e que, ao contrário do que acontece em ferramentas como o Eclipse ou o Visual Studio, não permite acesso à fonte de código em HTML e em que todo o desenho é controlado via CSS. Usar esta interface de desenho (Figura 17), é no fundo fazer uso das suas capacidades de "drag-and-drop", ou seja fazer uso de arrastamento de componentes para as páginas e dessa forma ir construindo a aplicação que se pretende. Sempre que o desenvolvedor se encontra dentro de uma página relacionada com uma base de dados, tem acesso às entidades desta através da árvore exploratória acessível do lado direito, e pode assim criar campos que manipulam as mesmas, bastando arrastar os atributos para o interior da página, resultando na criação automática de campos de acordo com os atributos dos mesmos na base de dados (como exemplo típico, arrastar uma entidade com relação de um para muitos resulta na criação automática de uma drop-down box para escolha).

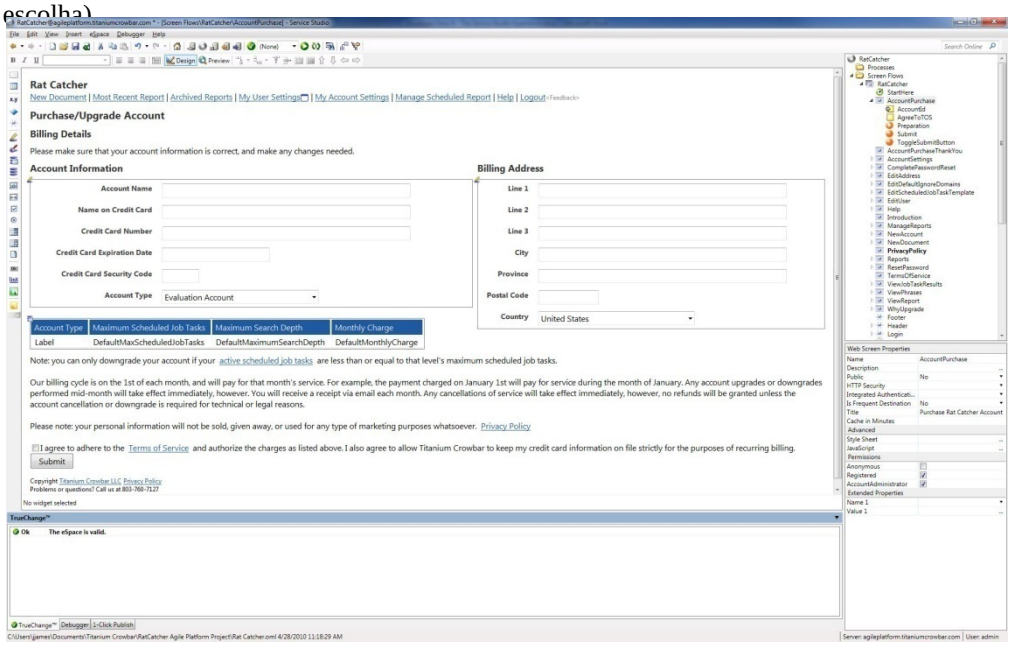


Figura 17: Implementação de uma página através da Interface de Desenho da Plataforma OutSystems

É indiscutível que este tipo de interfaces e respectivas tarefas, permitem aumentar de forma substancial a eficiência e simplicidade do desenvolvimento de aplicações. No entanto, para além de não ser possível o acesso directo ao HTML, a principal falha ao nível deste sistema, é o facto das páginas não se basearam num sistema em que todas as páginas podem herdar as mesmas características de um mesmo template, obrigando o desenvolvedor a especificar em todas a mesma estrutura visual.

Também o facto de não ser escrito qualquer pedaço de código, requer alguma habituação e aprendizagem por parte de desenvolvedores habituados a outro tipo de plataformas, sendo que o mais próximo que se fica de programar é ao escrever condições ao nível da interface de modelação de processos.

De realçar a forma característica como toda a relação entre campos da aplicação e campos de bases de dados ser feita através de simples mapeamento ao nível da interface gráfica, não havendo qualquer configuração manual para fazer corresponder atributos a atributos, ou formatos de dados a formatos de dados. Sempre que um campo ligado a uma base de dados é modificado a um dos níveis, as necessárias alterações são automaticamente reflectidas no outro nível, evitando assim qualquer inconsistência ao nível dos dados, e consequentemente qualquer erro ao nível da aplicação relacionado com o mapeamento de informação em bases de dados.

Ao nível do Service Studio, é sempre possível de verificar que uma aplicação irá compilar correctamente, no entanto a existência de um outro sistema designado "TrueChange" permite saber em tempo real quando poderá ocorrer um problema, tal como se encarrega de forma automática de propagar alterações em variáveis a toda a aplicação, isto é, se o tipo de uma variável é modificado, todas as referências a essa variável são também modificadas, e são indicados erros que possam ter resultado dessa modificação.

Outro dos pontos mais simples da plataforma, e que já foi várias vezes referidos na descrição da mesma, é o sistema de produção de uma aplicação que através de um simples clique permite que a aplicação seja produzida em .Net ou Java.

Como forma de demonstrar possíveis produtos resultantes do desenvolvimento nesta plataforma temos os casos típicos dos sítios Web do "Turismo de Portugal" e da "TNT" (Figura 18):

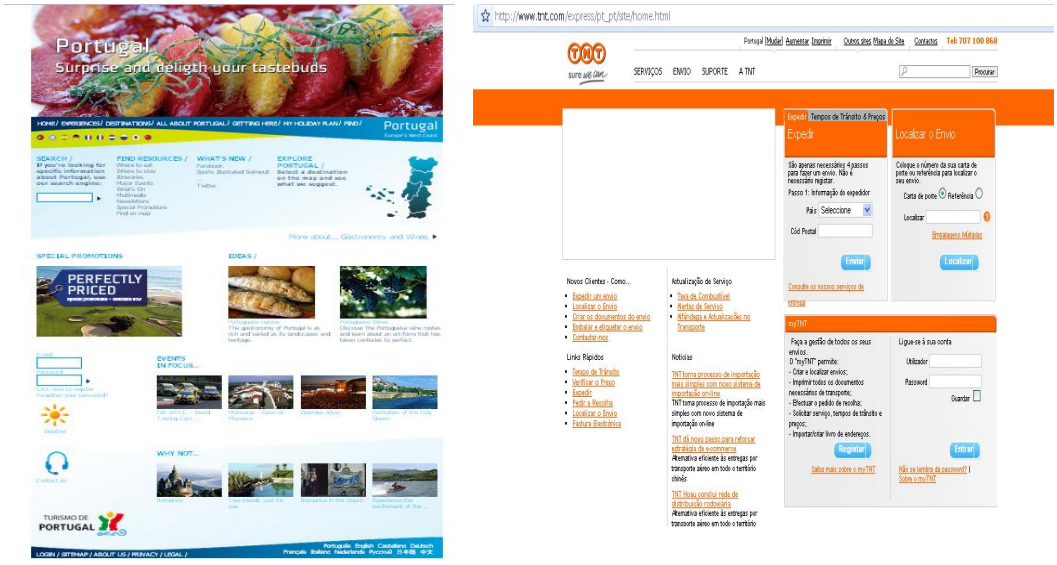


Figura 18: Aspecto de dois sítios Web desenvolvidos com a Plataforma Ágil da OutSystems

Assim, e concluindo sobre a sua utilização, o Service Studio da plataforma de desenvolvimento da OutSystems é uma ferramenta muito fácil e poderosa de usar e representa um passo revolucionário no que diz respeito a plataformas rápidas de desenvolvimento, ao excluir por completo a programação do desenvolvimento de aplicações e serviços Web. Sobre as vantagens do seu uso no CI-FCUL relativamente à plataforma adoptada, reflecte-se no capítulo seguinte.

4.2.3 Discussão sobre Adopção da DIF2.0 como Plataforma de Desenvolvimento no CI-FCUL

Após a exposição das características e da demonstração do funcionamento da plataforma de desenvolvimento DIF2.0 adoptada pelo CI-FCUL, bem como das plataformas que lhe serviriam de alternativa, procede-se agora com a explicação pormenorizada das razões que levaram à adopção da Plataforma de Desenvolvimento DIF2.0, para depois se concluir sobre as vantagens e desvantagens da adopção de cada uma das plataformas de desenvolvimento estudadas como alternativa, relativamente a esta plataforma da Digitalis.

Razões de Adopção da DIF2.0 no CI

Após descrição de toda a plataforma DIF2.0 e depois de ser possível ter uma ideia das suas componentes e características, emerge a necessidade de justificar a sua adopção pelo Centro de Informática da FCUL.

Características Técnicas Importantes

Naturalmente que as suas características são o principal impulsionador desta mesma opção, no entanto e para uma ideia mais exacta da forma como estas representam mais-valias para o CI convém realçar as que mais importam. Assim sendo, destacam-se as seguintes características:

Organização hierárquica de funcionalidades por entidades: Ao implementar funcionalidades e serviços a partir de uma estrutura hierárquica de classes definida desde o início do desenvolvimento, possibilita-se:

- Uma maior organização estrutural da informação e divisão desta por diversos níveis informativos, sendo possível separar funcionalidades, de serviços e de simples funções acessórias.
- Acesso mais facilitado a código que implementa diferentes serviços e funcionalidades e consequente reutilização destes em outros projectos evitando esforço e tempo de implementação.

Geração Automática de Código: Ao resumir algum do desenvolvimento de funcionalidades à escrita de pequenos pedaços de texto que possibilitam a geração automática do código que as implementa, permite-se:

- Uma poupança de esforço no desenvolvimento de funcionalidades mais exigentes e uma diminuição da complexidade programática de uma funcionalidade ou mesmo serviço/aplicação.

- Um aumento de eficácia e de facilidade na implementação, não só acelerando o processo como também evitando e reduzindo erros na implementação de funcionalidades automaticamente geradas.
- Maior uniformidade na implementação de funcionalidades semelhantes em diferentes projectos.

Implementações Modulares e Uso de componentes Web 2.0: O facto de toda a arquitectura desta plataforma se basear na existência de diversos módulos integrados numa base geral, reflecte-se também numa implementação baseada em módulos de funções que implementam diferentes funcionalidades ou na integração e uso de componentes Web 2.0, permitindo:

- Uma maior reutilização de componentes e funcionalidades, uma vez que é escusado desenvolver novamente uma funcionalidade quando esta já existe sobre a forma de um módulo facilmente integrável.
- Diminuição de esforço e de erros no desenvolvimento, com a possibilidade de integração de funcionalidades já desenvolvidas e que exigiriam grande esforço de implementação.
- Maior uniformidade na implementação de funcionalidades semelhantes em diferentes projectos.

Factores Institucionais

Para além de factores relacionados com as características da plataforma de desenvolvimento, existem outros que jogaram a favor da sua adopção, e que se prendem essencialmente com características das ferramentas já utilizadas e desenvolvidas para a instituição. Assim destacam-se os seguintes pontos:

- Grande número de aplicações e serviços desenvolvidos pela empresa Digitalis para a faculdade (sistemas de inscrições, sistema de gestão de alunos, professores, disciplinas, exames, calendários, etc.) e sua possibilidade de integração com a plataforma DIF2.0 desenvolvida pela mesma empresa, confere maior fiabilidade ao produto aos olhos da FCUL.
- Essa mesma fidelização com a empresa, e satisfação ao nível de suporte oferecido durante anos de utilização dos seus produtos, faz antever um suporte idêntico ao nível de utilização da plataforma de desenvolvimento. Sendo grande a vantagem de um suporte constante, isso jogou em grande medida a favor da adopção da ferramenta.
- Contactos exploratórios com a empresa no âmbito de adopção da plataforma desencadearam a possibilidade de uma formação inicial da equipa de desenvolvimento do CI na mesma, oferecendo a possibilidade de experimentar a plataforma e de diminuir a curva de aprendizagem típica neste tipo de sistemas. Mais uma vez, a disponibilidade desta empresa para uma formação deste género, deixou antever um suporte eficaz e constante por parte da mesma, no uso da plataforma. Este foi também um ponto que pesou muito a favor da adopção da DIF2.0 2.0.

Destes dois conjuntos de factores destaca-se como mais importante o segundo, porque na realidade foram os factores institucionais que ditaram a adopção da DIF2.0 como plataforma de desenvolvimento

para o CI-FCUL. Essa grande ligação entre a empresa desenvolvedora da plataforma e todos os mecanismos usados pela Faculdade, foi o principal factor responsável pela não execução de um estudo mais elaborado sobre alternativas a esta ferramenta de forma prévia à sua adopção, e é com origem nesta situação, que se desencadeia uma relevante discussão sobre alternativas futuras de desenvolvimento no Centro de Informática, com base na utilização ou não da plataforma em causa (discussão esta que pode ser encontrada no penúltimo capítulo deste documento).

Problemas Identificados na Adopção da Plataforma

Da análise desta Plataforma de Desenvolvimento resulta naturalmente a identificação de problemas verificados com a adopção da ferramenta por parte do CI-FCUL, que convém passar a enumerar antes mesmo de se partir para o estudo das vantagens de adopção de alternativas a esta ferramenta:

- Sabendo-se que as principais razões para adopção desta Plataforma, estiveram relacionados mais com ligações institucionais entre a FCUL e a empresa desenvolvedora da Plataforma, e que a mesma Plataforma não foi alvo de atenta experimentação antes de ser de facto adoptada (o período de uma semana de treino realizado sobre plataforma, marcou já o início da sua adopção e não um teste para verificação das suas qualidades), percebe-se que não houve qualquer comparação com outras plataformas existentes de forma gratuita (ou semi-gratuita) na Web, e que por isso se confiou cegamente numa plataforma que na realidade não se conhecia suficientemente bem, para se optar pela sua adopção.

- O facto da plataforma DIF2.0 ser um produto open-source e portanto não estar completamente finalizado, e sempre propenso a novas extensões, ou à descoberta de problemas originalmente não identificados, representa um ponto negativo da sua adopção. Este tipo de problemas ocorreria com qualquer ferramenta open-source que fosse adoptada.

- O facto de não haver grande comunidade de utilizadores que utilizam a plataforma e o facto do sítio oficial não conter grande informação de suporte ao desenvolvimento - a informação existente é claramente insuficiente para a aprendizagem da plataforma sem suporte directo da equipa que a desenvolveu – fazem com que a aprendizagem da plataforma seja muito difícil e muito demorada, e baseada num constante pedido de esclarecimento de dúvidas à equipa responsável pelo desenvolvimento da mesma.

CakePHP - Vantagens e Desvantagens na Adopção Relativamente à DIF2.0

A melhor forma de perceber se esta plataforma seria ou não uma alternativa válida à Digitalis Internal Framework é identificar os pontos a favor e os pontos contra a adopção da mesma no Centro de Informática da FCUL, evidenciando se esses pontos representam vantagens ou desvantagens sobre a plataforma da Digitalis.

Pontos a favor da adopção

1. Quando em comparação com a DIF2.0, ao nível de simplicidade de utilização, o CakePHP é bastante superior e bastante mais gratificante, naquilo que permite atingir com menos esforço. Sendo que ambas apresentaram simplificação na implementação do mesmo tipo de funcionalidades, acontece que no caso da primeira plataforma esse uso de funcionalidades é muito mais difícil de entender e necessita de um maior número de passos, ou de uma maior complexidade de execução e codificação.

2. Em termos de facilidade de aprendizagem, também esta plataforma é superior à DIF2.0, uma vez que o esforço dispendido a início para realização de tarefas simples, isto é, para implementação de funcionalidades básicas, é bastante inferior e a programação é mais gratificante e fácil de memorizar que o tipo de programação efectuada na plataforma que foi de facto adoptada.

3. Também no que respeita à comunidade existem grandes diferenças entre as duas plataformas. Sendo que a DIF2.0 é um produto ainda numa fase mais inicial de desenvolvimento que a plataforma CakePHP, a sua comunidade de utilizadores é claramente mais reduzida que as da segunda, plataforma esta já com várias versões lançadas na Web e com uma comunidade de utilizadores bastante grande e diversificada.

4. A superioridade evidenciada em termos de comunidade de utilizadores, reflecte-se também em termos de suporte e de quantidade, qualidade e diversidade de documentos que servem de apoio à utilização de ambas as plataformas. Sendo que a DIF2.0 contém como documentação apenas o sítio Web oficial da plataforma, e que essa informação se apresenta até à data bastante incompleta, enquanto para a plataforma CakePHP é possível encontrar muita informação por toda a Internet, incluindo vários vídeos produzidos pela sua comunidade de utilizadores, guias de utilização oficiais e não oficiais, e no geral uma documentação bastante rica e completa sobre pormenores de utilização e especificidades da sua arquitectura. Assim sendo, neste ponto, CakePHP mostra-se também imensamente superior.

Pontos contra a adopção

1. Do ponto de vista da capacidade de configuração da plataforma, o CakePHP possibilita apenas opções muito básicas, relacionadas essencialmente com a informação presente em bases de dados e com parâmetros de validação de formulários, enquanto a DIF2.0 servindo-se do Maven, possibilita grande nível de configuração de todos os pormenores relacionados com a plataforma e com o desenvolvimento nesta.

2. Outro ponto em que possivelmente o CakePHP poder-se-á revelar ligeiramente inferior é o de costumização ou criação de novas funcionalidades, bem como na implementação de mecanismos de geração automática de código (embora fosse preciso uma análise mais profunda, e uma experimentação mais que curiosa da plataforma para confirmação deste facto). Fica-se com a ideia que no CakePHP, estando já desenvolvido um número grande de funcionalidades, não é

oferecida grande margem para implementação de novas funcionalidades por parte da sua comunidade de utilizadores, enquanto no caso da DIF2.0 e como resultado de estar ainda agora a iniciar a sua expansão e de haver ainda grande quantidade de funcionalidades a necessitar de implementação, a forma como é possibilitada a extensão da plataforma por parte dos seus utilizadores é mais aberta.

3. O ponto mais óbvio de inferioridade da plataforma programada em PHP em relação à plataforma em Java, é aquele que já foi previamente referido aquando da descrição da DIF2.0, o facto de com essa plataforma haver um relacionamento mais directo com os seus criadores e portanto uma possibilidade maior de resolução de problemas e de personalização do produto.

4. Por fim, o facto de haver à partida uma maior facilidade de integração da plataforma DIF2.0 com o conjunto de serviços utilizados na instituição em causa, e também desenvolvidos pela mesma empresa responsável pelo desenvolvimento da plataforma, representa uma vantagem óbvia da utilização preferencial da DIF2.0 sobre o CakePHP.

Conclusão

Da análise dos pontos a favor e a desfavor da utilização da plataforma CakePHP sobre a DIF2.0, conclui-se que a primeira seria uma aposta mais segura que a segunda, em termos de utilização e desenvolvimento imediato, enquanto a segunda será a melhor alternativa na perspectiva de que no futuro irá atingir uma maior consistência e garantir um maior suporte à aprendizagem.

OutSystems – Vantagens e Desvantagens na Adopção Relativamente à DIF2.0

A melhor forma de perceber se esta plataforma seria ou não uma alternativa válida à Digitalis Internal Framework é identificar os pontos a favor e os pontos contra a adopção da mesma no Centro de Informática da FCUL, evidenciando se esses pontos representam vantagens ou desvantagens sobre a plataforma da Digitalis.

Pontos a favor da adopção

1. A principal vantagem de utilização desta plataforma em relação à DIF2.0 é o facto de, sendo uma plataforma muito poderosa e que permite obter melhores resultados finais que a plataforma da Digitalis, ser também uma plataforma de maior simplicidade de uso e que permite aumentar a rapidez no desenvolvimento de aplicações, bem como reduzir a quantidade de erros típicos num desenvolvimento. Para isto, muito contribui a ausência total de codificação de funcionalidades num desenvolvimento completamente baseado numa interface de desenho muito completa.

2. Em termos de automatismos e de poupança de esforço ao nível do desenvolvimento, esta plataforma também ultrapassa claramente a plataforma DIF2.0, nomeadamente no que diz respeito à simplicidade de processos com que é mapeada a relação entre campos nas aplicações (formulários) e

entidades nas bases de dados, bem como na forma como são identificadas inconsistências e evitados erros aquando de alterações em variáveis ou componentes no desenvolvimento.

3. Mesmo ao nível do processo de aprendizagem, de todo o funcionamento e uso desta plataforma, existirá uma maior facilidade de interiorização dos diferentes processos do que acontece nos processos de aprendizagem necessários ao início de desenvolvimento na DIF2.0 e sobretudo a curva de aprendizagem e o nível de gratificação com o aparecimento de resultados é bastante mais favorável à ferramenta da OutSystems.

Pontos contra a adopção

1. Um dos principais pontos contra a adopção desta plataforma em detrimento da Plataforma de Desenvolvimento da Digitalis, é o facto da adopção desta plataforma representar uma radical mudança no processo de desenvolvimento do CI-FCUL, e por isso poder ser alvo de pontos de resistência à mudança muito fortes no interior da instituição. O desconhecimento desta forma de desenvolvimento que exclui por completo os anteriores hábitos de codificação de funcionalidades, representa um factor de risco que a FCUL poderá não estar disposta a correr.

2. O ponto mais decisivo contra a adopção da plataforma da OutSystems por parte do CI-FCUL é o facto de este produto não ser gratuito como a plataforma DIF2.0, exigindo um pagamento mensal dependente da quantidade de serviços, utilizadores e desenvolvedores que farão uso da plataforma (existe uma versão gratuita mas que não serve para as necessidades da FCUL). Esse custo não sendo muito grande, representa uma grande desvantagem em relação a plataformas open-source inteiramente gratuitas e que oferecem basicamente o mesmo tipo de suporte.

3. Tal como é paga, a plataforma não permite também acesso ao seu código e muito menos ao código das funcionalidades desenvolvidas através das suas interfaces de implementação de serviços. Este factor para além de requerer alguma habituação, representa uma limitação para o grupo de desenvolvedores do CI-FCUL uma vez que não permite uma costumização das funcionalidades existentes (em mais do que a própria interface de desenvolvimento da plataforma permite) e não possibilita alteração de funcionalidades de forma livre através do acesso ao código que as implementa (este código apenas é gerado aquando da produção da aplicação).

4. Por último, e quase apenas como reparo, o facto de no desenvolvimento usando esta plataforma ser necessário a definição constante da mesma estrutura de apresentação em todas as páginas no mesmo projecto, devido a ausência uma definição de apresentação baseada em templates, representa uma pequena desvantagem em relação ao tipo de apresentação gráfica disponibilizada pela DIF2.0 (no entanto e como já disse, essa desvantagem é mínima).

Conclusão

Da análise dos pontos a favor e a desfavor da utilização da plataforma Ágil de Desenvolvimento da Outsystems sobre a Plataforma Rápida de Desenvolvimento da Digitalis DIF2.0, conclui-se que a

primeira seria uma aposta revolucionária e por isso arriscada de desenvolvimento, mas permitiria acelerar em grande medida o desenvolvimento de Serviços e Aplicações por parte do CI-FCUL, ao mesmo tempo que tornaria o processo de desenvolvimento mais facilitado para qualquer desenvolvedor (incluindo personalidades sem qualquer experiência em programação), enquanto a segunda constitui uma opção mais barata e menos arriscada mas muito menos eficiente. É de concluir, que a Plataforma de Desenvolvimento da OutSystems representa tudo aquilo que se pode querer numa plataforma cujo objectivo é acelerar, simplificar e tornar mais eficiente o desenvolvimento de aplicações e serviços Web, e que o seu único inconveniente acaba por ser os custos que a sua adopção acalataria para o CI-FCUL.

4.3 Considerações Finais

De todo este capítulo existem uma série de considerações a fazer no que diz respeito tanto a ferramentas de gestão de conteúdos como no que diz respeito a plataformas de rápido desenvolvimento de aplicações.

O primeiro ponto de destaque, relaciona-se com o facto de haver uma série de características que se procuram num Gestor de Conteúdos para que este seja adoptado pelo CI-FCUL. Como parte dessas caractetísticas, destacam-se a apresentação flexível de conteúdos, a instauração de processos de fluxos de acções relativamente simples, a possibilidade de reutilização de conteúdos, e sobretudo a possibilidade de gestão e uso simplificado da ferramenta para criação e modificação de conteúdos, de uma forma orientada a editores em que aquilo que está visível ao utilizador é exactamente o mesmo que ficará visível ao nível do conteúdo (editores WYSISYG).

Como segundo ponto de relevo, temos o estudo do gestor de conteúdos mais apropriado e indicado à adopção no CI-FCUL, em que como ponto de partida foi considerado que como condições primordiais teria que ser uma ferramenta que possibilitasse o uso dos conteúdos desenvolvidos por parte de outras ferramentas de desenvolvimento, e em segundo lugar teria obrigatóriamente que ser uma ferramenta programável em JAVA ou PHP e geradora de código PHP.

Após uma análise elaborada de vários CMS, destacaram-se duas ferramentas distintas, o Drupal e o Joomla, como as mais eficazes para todo o tipo de tarefas e características pretendidas. Destas, escolheu-se a primeira opção devido à sua superioridade no que diz respeito à facilidade de uso das interfaces e menus, devido à grande capacidade de aprendizagem que oferece, devido ao grande suporte oferecido em termos de comunidade e extensões da ferramenta e sobretudo devido à sua grande capacidade para gestão e desenvolvimetno de conteúdos. Em todos estes aspectos o Drupal se revelou de alguma forma superior ao Joomla, e foi por isso o escolhido.

No que diz respeito à segunda parte deste capítulo, e portanto à escolha de uma plataforma de desenvolvimento rápido de aplicações, não houve a início qualquer estudo como no caso do gestor de conteúdos, uma vez que à partida para este projecto estava já escolhida por parte do CI-FCUL (e seus

responsáveis) a ferramenta deste género a utilizar, uma plataforma denominada DIF.2.0. que como principais razões de adopção apresenta o facto de ser desenvolvida pela mesma empresa que desenvolveu a maior parte dos serviços usados na Faculdade. Assim, o estudo de alternativas foi mais breve, e realizado no sentido de perceber quais as reais capacidades da plataforma adoptada relativamente a outras existentes.

No contexto deste estudo foram analisadas duas ferramentas denominadas CakePHP e OutSystems Framework, e da análise dos pontos positivos e negativos das duas relativamente à DIF.2.0, concluiu-se que: a primeira, como ferramenta de maior simplicidade de uso e aprendizagem e uma vez que oferece um maior suporte por parte de uma comunidade mais extensa de utilizadores, seria uma aposta mais segura que a plataforma adoptada, em termos de utilização e desenvolvimento imediato de aplicações, deixando-se no entando reservas de que no futuro, e com uma melhoria do suporte e da consistência da ferramenta da Digitalis a sua solução passe a ser a melhor alternativa; a segunda como ferramenta automatizadora do desenvolvimento e devido à sua grande capacidade de desenvolvimento seria uma aposta que permitiria acelerar em grande medida o desenvolvimento de serviços, mas que no entanto seria uma aposta muito arriscada e revolucionadora de todos os métodos de desenvolvimento no CI-FCUL, devido à sua completa ausência de programação. Da conclusão do estudo das alternativas conclui-se que embora a plataforma DIF2.0 seja uma ferramenta capaz para o desenvolvimento das aplicações e serviços pretendidos não será a melhor opção, e que com uma estudo mais abrangente de outras alternativas, poder-se-ia chegar a uma alternativa mais consistente, segura e simples.

Após este capítulo, impõe-se a realização de estudos de uso das diferentes ferramentas que se pretende adoptar, bem como a análise de qual a melhor forma de desenvolvimento conjunto fazendo uso das duas aplicações ou de apenas uma para atingir certos objectivos de desenvolvimento. O capítulo seguinte, abordará estas questões.

Capítulo 5 Uso de Ferramentas Para Desenvolvimento Futuro de Aplicações

Após a análise cuidada das plataformas a utilizar no desenvolvimento futuro de aplicações e serviços no CI-FCUL, interessa introduzir essas ferramentas na metodologia definida e estudar qual a melhor forma de desenvolvimento com base no seu uso conjunto ou isolado. Para chegar a estas conclusões, nada é melhor que a realização de casos de estudo que compreendem a utilização e a integração das ferramentas em casos práticos de desenvolvimento, e que o estudo do esforço dispendido para realização das tarefas ou para a implementação das funcionalidades mais típicas de encontrar num serviço Web.

Assim sendo, neste capítulo estudam-se duas formas de desenvolvimento, sendo a primeira, o desenvolvimento conjunto utilizando a plataforma rápida de desenvolvimento DIF2.0 e o gestor de conteúdos Drupal, e sendo a segunda o desenvolvimento baseado apenas no mesmo gestor de conteúdos. É importante dizer previamente, que esta possibilidade de desenvolvimento fazendo uso de apenas uma das ferramentas acabou por surgir na sequência da realização dos próprios casos de estudo referidos.

5.1 Desenvolvimento Conjunto com DIF2.0 e Drupal

Para que um gestor de conteúdos como o Drupal e uma plataforma rápida de desenvolvimento como a DIF2.0, coexistam no processo de desenvolvimento terá de existir sempre uma clara distinção no tipo de tarefas levadas a cabo por cada uma delas:

O Gestor de Conteúdos Web terá como principais funções: a manutenção, actualização e criação de páginas com conteúdo estático; a personalização, diversificação e possivelmente a criação de templates de apresentação de conteúdos; a manutenção de menus de navegação e ligações para outros conteúdos; a publicação de conteúdos regulares; e servirá como repositório de dados fazendo a sua manutenção simultânea e podendo aplicar sobre estes simples fluxos de aprovação de conteúdos. O uso do CMS será assim da responsabilidade de um conjunto de pessoas que, não participando em processos de desenvolvimento mais complexos, utilizarão as ferramentas embebidas no CMS para edição, criação e manutenção de componentes e conteúdos estáticos.

A plataforma de desenvolvimento DIF2.0 terá como principais funções: a manutenção, actualização e criação de páginas com conteúdo dinâmico (formulários e todo o tipo de conteúdo com comunicação com as bases de dados do CI-FCUL); a invocação de conteúdos desenvolvidos e mantidos pelo Gestor de Conteúdos e respectiva integração com os restantes conteúdos desenvolvidos nesta plataforma; o desenvolvimento ou edição de templates de apresentação de conteúdos. Em suma, a plataforma de desenvolvimento rápido de aplicações terá a função de ser responsável pelo desenvolvimento geral das aplicações e serviços do Centro de Informática, sendo capaz de importar os conteúdos estáticos mantidos pelo CMS e os integrar com os restantes conteúdos dinâmicos de forma semi-automática, não tendo que haver a preocupação para os programadores da DIF2.0 de desenvolverem conteúdos estáticos regularmente, preocupando-se apenas com conteúdos mais complexos e dinâmicos.

A forma de integração destes dois tipos de ferramentas é de seguida descrita ao pormenor, seguindo-se um estudo de uma implementação conjunta e a forma como as duas se integram na metodologia de trabalhos definida.

5.1.1 Integração do Drupal com Plataforma de Desenvolvimento DIF2.0

Após a descoberta do sistema de gestão de conteúdos, mais apropriado para adopção no Centro de Informática da FCUL, interessa garantir que esta ferramenta é integrável com outras ferramentas de trabalho a utilizar no futuro no desenvolvimento rápido de serviços Web neste local. Como já referi anteriormente neste relatório, tendo as duas ferramentas de desenvolvimento objectivos e funções diferentes, é essencial que as duas possam coexistir num mesmo serviço, fazendo-se uso de conteúdos estáticos produzidos pelo CMS em serviços desenvolvidos através da plataforma de desenvolvimento, tendo para isso que se passar por um processo de importação e exportação de conteúdos entre as duas ferramentas. Sabemos que para que as duas plataformas de desenvolvimento coexistam, tem que haver

uma separação e distinção no que diz respeito a conteúdos desenvolvidos, templates de apresentação, gestão de informação e utilizadores das ferramentas, e são essas diferenças que passo a identificar:

Conteúdos Drupal VS Conteúdos DIF2.0

Existe uma clara diferença no tipo de conteúdos e dados que podem ser produzidos por um gestor de conteúdos Web como o Drupal quando em comparação com uma plataforma de desenvolvimento como a DIF2.0.

O grande objectivo com a adopção de um CMS como o Drupal, seria sempre o de simplificar a produção de tudo o que é conteúdo estático, isto é, informação sobre a forma de textos, imagens, vídeos, animações, gráficos, tabelas, tal como ainda os próprios menus de navegação e os índices de informação, e garantir que todo este tipo de informação pode ser criado por utilizadores comuns, sem grandes conhecimentos de qualquer linguagem de programação. Como exemplo, consideremos a implementação de uma página principal num sítio Web, que vai sendo regularmente actualizado com as últimas notícias da instituição. Estas actualizações, através do Drupal, são feitas de uma forma muito simples e por intermédio de um utilizador comum, que tem como única tarefa a escrita das notícias, como se tivesse a escrever simples documentos de texto e sem ter que programar qualquer pedaço de HTML ou PHP (o que acontece na actualidade). Tal como neste exemplo, todo e qualquer conteúdo que não necessite de aceder a uma tabela de uma base de dados para escrever ou carregar informação, ou que não necessite de ser encaminhada para endereços de email ou carregada/inserida em formulários Web, deve ser gerida pelo gestor de conteúdos Drupal e por utilizadores comuns.

Para informação entendida sobre a forma de um serviço, ou seja, para informação que necessite de facto de algum carregamento de dados de uma base de dados ou de conteúdos externos, ou que tenha de desencadear um qualquer procedimento de verificação, actualização ou inserção de dados e que precise por essa razão de ser necessariamente programada sobre a forma de código numa qualquer linguagem (neste caso em Java/PHP), é utilizada a plataforma de desenvolvimento rápido DIF2.0 e são codificados serviços sobre a forma de funções por utilizadores com devidos conhecimentos programáticos.

Havendo esta clara distinção entre as tarefas e respectivas ferramentas de desenvolvimento, interessa no entanto referir ainda, que o Drupal pode ser usado também para implementação de conteúdos dinâmicos, uma vez que, devido à grande quantidade e vastidão de módulos instaláveis que apresenta, permite implementar, sem grande dificuldade, funcionalidades que envolvem este tipo de conteúdos. Essa implementação, no entanto, deve ser cuidada e ter em conta diversos factores - como por exemplo os factores relacionados com a ligação às diferentes bases de dados -, e por isso mesmo deve ser levada a cabo, idealmente, por utilizadores com conhecimentos programáticos e que possam fazer uma “ponte” entre estes conteúdos desenvolvidos e os conteúdos do mesmo tipo desenvolvidos na DIF2.0. No entanto, maior elaboração sobre o desenvolvimento de conteúdos deste género em Drupal é feita na segunda parte do capítulo em “Desenvolvimento Unicamente em Drupal”.

Maior pormenorização sobre a forma prática como conteúdos dinâmicos e estáticos podem coexistir no mesmo serviço/sítio Web, é feita de seguida.

Integração Drupal com DIF2.0

Uma vez que, num serviço Web, existem normalmente os dois tipos de conteúdos já descritos, é naturalmente necessário que se possam conciliar as duas ferramentas num mesmo resultado final. Isto faz-se através do uso e importação de conteúdos implementados pelo Drupal para a DIF2.0, para que não seja necessária grande gestão de actualizações ao nível da plataforma de desenvolvimento, ou seja, quer-se que esta importação de dados seja de alguma forma dinâmica, e que a actualização de dados estáticos no CMS resulte na devida alteração ou inserção de dados nos conteúdos importados pela DIF2.0.

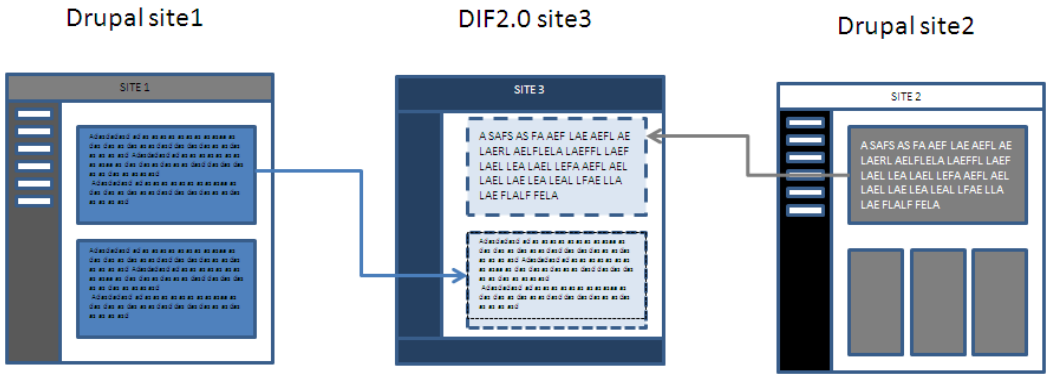


Figura 19: Integração de Conteúdos Estáticos do Drupal na DIF2.0

Como a figura sugere, o objectivo principal da integração de conteúdos Drupal em serviços desenvolvidos pela DIF2.0 é, para além da gestão independente desses conteúdos, a capacidade de poder importar diversos conteúdos de vários Sítios Web desenvolvidos em Drupal e de os englobar num novo Serviço, apresentando esses conteúdos de forma fiel à sua origem ou alterando a sua apresentação possibilitando uma melhor contextualização no novo serviço desenvolvido (como é referido em "Templates de Apresentação Não Uniformizados").

Aspectos Técnicos da Integração Drupal – DIF2.0

Como forma de provar a integração entre os dois produtores de conteúdos (Drupal e DIF2.0) explico os pormenores técnicos (já testados) que a possibilitam:

Consideremos a existência de duas partes específicas na integração. A primeira parte é formada pelo pedido do conteúdo pretendido por parte da DIF2.0 à página pretendida gerada pelo Drupal, enquanto a segunda parte é formada pela resposta da DIF2.0 ao cliente browser que em primeiro lugar invoca o conteúdo, resultando na respectiva apresentação do mesmo:

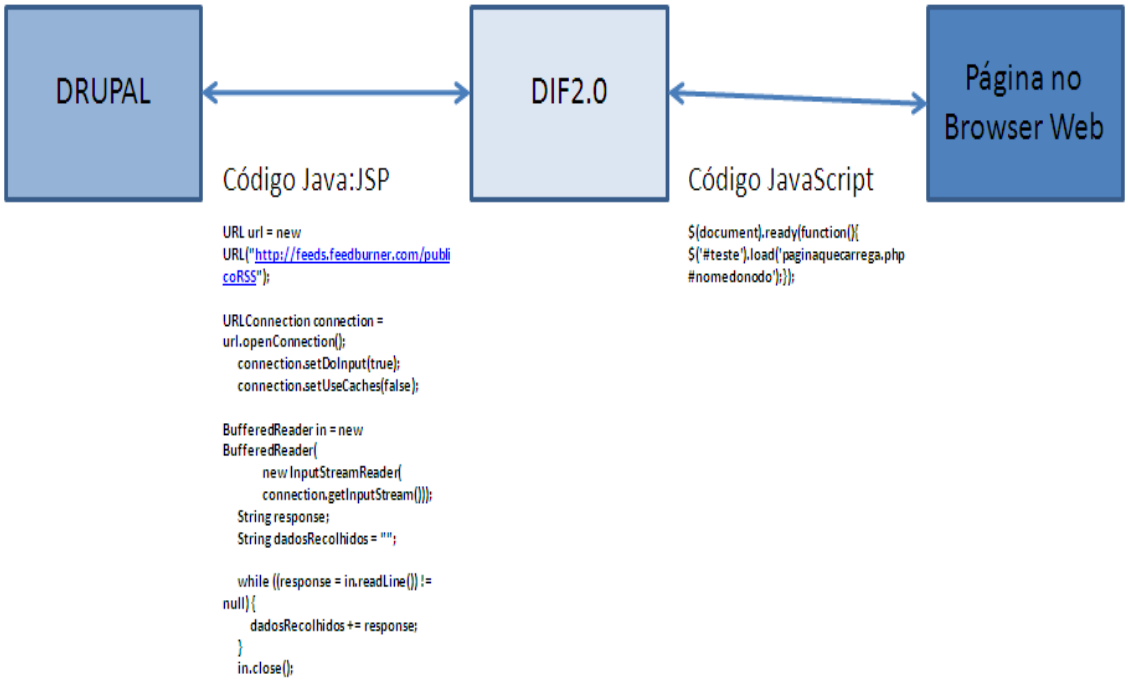


Figura 20: Integração entre DIF2.0 e Drupal e exemplo de código necessário à apresentação de conteúdos importados

Assim sendo, todo o processo se baseia no pedido por parte da DIF2.0 do conteúdo Drupal pretendido através do seu URL, após o qual a página recebida é guardada numa variável para se proceder ao seu tratamento e ao isolamento do conteúdo do(s) nó(s) de informação específico(s) pretendido(s), através da identificação de tags HTML que identificam cada pedaço de informação.

É assim possível, através de código semelhante ao que está na figura, a importação de qualquer nó de informação "fabricado" pelo Drupal (código JSP) tal como é possível a posterior configuração da forma como o nó é apresentado ao nível do cliente Web através da manipulação das suas propriedades através de código JavaScript. É aliás neste ponto, que se poderá inclusivamente configurar o acesso a ficheiros CSS que possibilitam a configuração do estilo dos conteúdos apresentados desta forma. Por fim, esclareço ainda que, uma vez que este código é executado de cada vez que a página do cliente browser é pedida, sempre que existir uma alteração ao conteúdo importado ao nível do Drupal, essa alteração também se irá reflectir ao nível do cliente, uma vez que a última vez que a página é importada esta já se encontra devidamente actualizada. Este é o exemplo prático que demonstra que a gestão de conteúdos estáticos pode ser feita num "lado" diferente da sua utilização, sem que o utilizador que o edita tenha que estar preocupado com isso, e sem que o programador que o utiliza tenha que se preocupar com essas ditas alterações de conteúdo.

No futuro, ir-se-ão realizar casos de estudo mais complexos deste exemplo, com posterior configuração de inteiros serviços implementados na ferramenta DIF2.0 que usam esta forma de integração para apresentação de conteúdos criados inteiramente no gestor de conteúdos Drupal. No

capítulo seguinte temos um caso de estudo de um serviço experimental de registo em conferências em que isto acontece.

Templates de Apresentação no Drupal e na DIF2.0

Uma das questões mais pertinentes de abordar quando se fala de existência de duas plataformas de desenvolvimento de conteúdos, prende-se com a forma de apresentação desses mesmos conteúdos, ou melhor com a uniformização ou não dessa apresentação.

Uniformização dos Templates de Apresentação:

A uniformização na apresentação de conteúdos será indispensável sempre que o desenvolvimento de um novo sítio tornar indispensável o mesmo tipo de formato de informação, isto é, sempre que for necessário o desenvolvimento de um novo sítio que tenha necessariamente de ter o mesmo aspecto que o serviço ou sítio que lhe dá origem, ou sempre que um novo sítio precise de ter o mesmo aspecto gráfico por uma questão de identificação institucional ou aspectos lógicos de relação com outros serviços. Neste aspecto, uma vez que, em ambas as plataformas de desenvolvimento de conteúdos, o código do template de apresentação se traduz numa série de linhas em linguagem HTML, a procura de uma uniformização no template de apresentação entre alguns sítios desenvolvidos no Drupal e serviços desenvolvidos na DIF2.0 poder-se-á traduzir num processo relativamente simples. Para que um sítio e um serviço nestas ferramentas tenham o mesmo tipo de apresentação e estilo é necessário que, antes de mais, seja possível utilizar os mesmos ficheiros CSS de estilos tal como é necessária a implementação do mesmo código HTML que define as diferentes zonas de apresentação de informação. Esta exigência obriga a que sempre que é criado um template de apresentação para uma plataforma seja criada a sua réplica para a outra plataforma também, garantindo que, caso haja necessidade, seja possível configurar dois sítios com o mesmo aspecto, independentemente da plataforma de desenvolvimento.

Como referi, visto ambas as plataformas implementarem o mesmo tipo de templates, esta integração adivinha-se relativamente simples de executar, faltando no entanto a sua concretização prática como prova destas impressões. Implementação esta, que não está englobada neste projecto, tendo sido lançado um concurso público por várias empresas para desenvolvimento de templates de apresentação para as duas plataformas.

Templates de Apresentação Não Uniformizados

Naturalmente, tal como em certos casos é necessária a uniformização na apresentação de informação, também existem casos em que interessará haver uma separação na forma de amostragem de informação no desenvolvimento em diferentes plataformas. Um exemplo disto, é o caso em que após implementação de um sítio Web inteiramente estático no CMS Drupal, se necessite de implementar um serviço através da DIF2.0 que possua funcionalidades mais complexas, mas em que devem ser reaproveitados conteúdos estáticos já desenvolvidos, e que se pretenda que os utilizadores-alvo

visualizem o serviço numa forma de apresentação completamente independente da que está presente no sítio Web originalmente responsável por esses mesmos conteúdos estáticos. Caso seja este o caso, a adopção de diferentes templates de apresentação bem como de diferentes CSS de estilo, nas diferentes plataformas, permitem essa heterogeneidade de aspectos. Estas duas possibilidades só se tornam possíveis devido à independência de templates existente no processo de integração de conteúdos entre as duas ferramentas de desenvolvimento.

Gestão e Utilizadores no Drupal e na DIF2.0

Na gestão de conteúdos existe uma separação clara de tarefas de gestão pelas duas ferramentas de desenvolvimento. O fácil uso, baseado em interfaces e opções automatizadas, do gestor de conteúdos Drupal, permite que este seja utilizado por quase qualquer pessoa após um período muito breve de aprendizagem. Para além disso, tendo um período de experimentação muito gratificante, permite que utilizadores que a princípio não tinham qualquer ideia de desenvolvimento de serviços Web, passem, num período relativamente curto de tempo, a ser responsáveis pela gestão de tudo o que é conteúdo estático apresentado pelos sítios Web desenvolvidos e geridos pelo CI-FCUL (libertando os programadores para as tarefas relacionadas com o desenvolvimento de serviços mais complexos).

No caso da DIF2.0 a gestão não pode ser feita por este tipo de utilizadores, uma vez que o desenvolvimento nesta ferramenta se baseia na codificação de funções que implementam serviços dinâmicos (comunicam com bases de dados de informação, ou estão dependentes de verificações, ou desencadeiam a activação de outros serviços, etc.). Assim sendo, utilizadores que normalmente eram responsáveis por quase toda a gestão de conteúdos independentemente da sua complexidade, passam a ser responsáveis pela implementação e gestão exclusiva dos conteúdos mais complexos, e passam a estar centrados apenas em tarefas de programação relacionadas com os diferentes serviços Web.

O desenvolvimento simultâneo nestas duas plataformas possibilita o reaproveitamento de informação e permite a existência única de um conteúdo ao nível de uma das plataformas, bem como a sua consequente reprodução em outros Sítios/Serviços Web, continuando este conteúdo a ser gerido na sua origem, pelos utilizadores que o criaram originalmente.

O quadro seguinte especifica os tipos de tarefas levadas a cabo por cada um dos diferentes tipos de utilizadores destas duas plataformas de desenvolvimento num desenvolvimento conjunto:

Tarefa de Gestão/Programação	Tipo de Utilizador:			Plataforma de Desenvolvimento
	Sem Conhecimentos Programáticos	Com Conhecimentos Programáticos Básicos	Programador	
Desenvolvimento de conteúdos estáticos	Sim	Sim	Não	Drupal
Desenvolvimento de conteúdos dinâmicos	Não	Sim	Sim	DIF2.0 (ou Drupal)
Gestão de conteúdos estáticos	Sim	Sim	Não	Drupal
Gestão de conteúdos dinâmicos	Não	Sim	Sim	DIF2.0 (ou Drupal)
Integração de conteúdos entre Drupal e DIF2.0	Não	Não	Sim	Drupal e DIF2.0
Implementação de Templates de Apresentação	Não	Sim	Sim	Drupal e DIF2.0 e Ferramenta de Desenho de *Templates*
Gestão de Templates de Apresentação	Não	Sim	Sim	Drupal e DIF2.0 e Ferramenta de Desenho de *Templates*

Tabela 7: Tipo de tarefas e utilizadores que as realizam

5.1.2 Caso de Estudo: "Marcação de Conferências"

Como um caso de estudo de desenvolvimento conjunto em duas ferramentas de implementação diferentes, "Marcação de Conferências" consiste no desenvolvimento de dois protótipos de sítios Web, um desenvolvido em Drupal e outro desenvolvido na DIF2.0. O primeiro consiste apenas num sítio Web informativo, que disponibiliza simples páginas de descrição de uma conferência e apresenta ligações para imagens ou para outras páginas internas, enquanto o segundo consiste num pequeno serviço experimental de reserva de salas e horas para a marcação de conferências e importa informação do primeiro sítio de forma a exemplificar a integração entre as duas ferramentas no sítio desenvolvido utilizando a DIF2.0.

Funcionalidades Implementadas

Para uma ideia compreensiva das capacidades de desenvolvimento conjunto nas duas plataformas foram implementadas várias funcionalidades típicas de um Serviço Web. Esta implementação foi sempre levada a cabo num ponto de vista experimental e por isso nunca vista como uma construção perfeita de funcionalidades no que diz respeito a formas de apresentação ou a total coerência de informação, e é por essa razão que chamamos aos sítios Web desenvolvidos, protótipos de aplicações e não aplicações definitivas. Estando esclarecido este ponto posso passar a apresentar os protótipos de uma forma geral, para depois me centrar em cada funcionalidade desenvolvida sobre estes:

Visão Geral das Aplicações

O primeiro sítio a que foi dado o título de "Conferência Experimental" foi desenvolvido sobre Drupal, e compreendeu apenas a construção de duas páginas de informação referentes à conferência em causa, e a definição de um template de apresentação básico em que se deu especial destaque ao cabeçalho como forma de identificação inequívoca do sítio. Para uma visualização geral do protótipo resultante, mostram-se de seguida imagens do sítio Web resultante (Figura 21):

Figura 21: Imagem do Sítio Web "Conferência Experimental"

O segundo sítio a que foi dado o título de "Marcação de Conferências na FCUL" foi desenvolvimento sobre DIF2.0, e compreendeu a construção de quatro páginas diferentes e uma base de dados, sendo que existe uma página principal constituída por textos informativos e uma área de login no serviço, outras duas páginas que manipulam informação presente numa base de dados através de formulários para registo e alteração de conferências (operações de inserção e actualização da base de dados) e uma quarta página que permite a visualização de todo o conteúdo inserido na base de dados. Destaque ainda para o facto de esta aplicação ter um template de apresentação em branco e sem qualquer informação adicional (não foi contemplado no estudo, a construção de um template na DIF2.0, como já foi antes referido). Para uma visualização geral do protótipo desenvolvido nesta plataforma de desenvolvimento, mostram-se de seguida imagens do sítio Web resultante (Figura 22):

Figura 22: Imagem do Sítio Web desenvlvido na DIF2.0

Implementação de Funcionalidades

Apenas verificando caso a caso as funcionalidades desenvolvidas em cada um dos protótipos de aplicação desenvolvidos, se pode ter uma melhor ideia do que é de facto o desenvolvimento nas plataformas que os implementam. Assim sendo:

"Conferencia Experimental": Páginas Informativas

A implementação de páginas simples contendo texto informativo, ou outro tipo de conteúdo estático ao nível deste protótipo e através da ferramenta Drupal, é uma tarefa muito simples de realizar e ao alcance de qualquer tipo de utilizador, requerendo inclusive muito pouco tempo de aprendizagem. Baseia-se assim na navegação no sítio criado pelo Drupal, e na selecção da opção "Criar conteúdo", e por conseguinte no preenchimento de campos de informação que entre "nome da página", "título" e "corpo" permitem criar uma página simples. A inserção de texto é feita em HTML, no entanto, e com a simples instalação de um módulo que possibilita a inclusão de um editor de texto simplificado (editor WYSIWYG) (Figura 23 e Figura 24) qualquer utilizador sem conhecimentos de HTML edita o "corpo" da sua página à semelhança de como edita um qualquer ficheiro de texto em Microsoft Word (inclusivamente na adição de imagens).

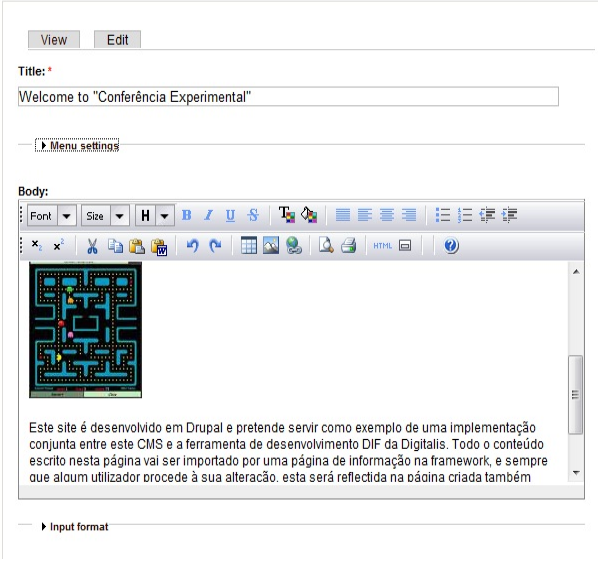


Figura 23: Imagem do editor WYSIWYG no Drupal

Desta forma, a criação das duas páginas informativas neste sítio Web foi realizada praticamente sem qualquer esforço por parte do desenvolvedor e apenas com base em uso simples de Drupal.

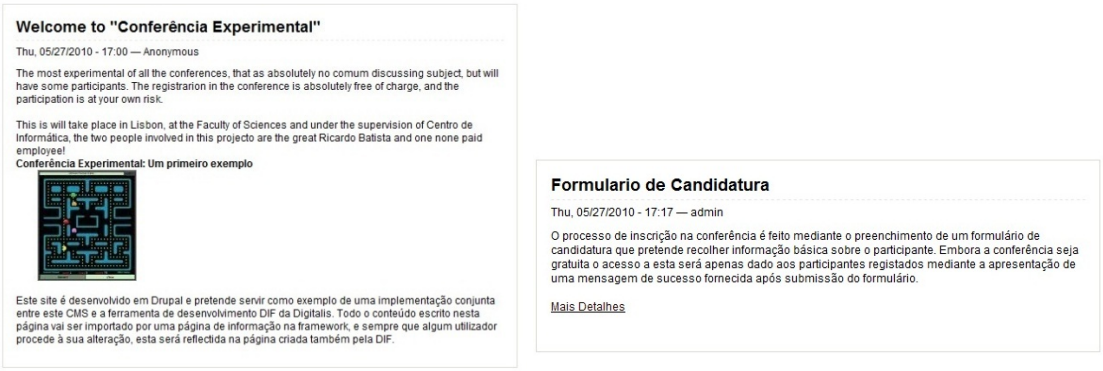


Figura 24: Conteúdos Informativos criados no Sítio "Conferência Experimental"

"Conferência Experimental": Template de Apresentação

O outro elemento alvo de atenção de desenvolvimento no Drupal e portanto no sítio "Conferência Experimental", é a implementação ou configuração de um template para definição da forma de expor o conteúdo do sítio Web e mostrar todas as suas componentes. No Drupal a forma de definição de um template, é semelhante à que era já usada no desenvolvimento no CI-FCUL, e faz-se através da codificação pura de ficheiros HTML, PHP e CSS. No entanto e como existe grande número de modelos de templates de apresentação fornecidos gratuitamente no site que oferece suporte ao Drupal, foi aproveitado um modelo simples de apresentação, que foi posteriormente alterado em termos essencialmente de cabeçalho (construção de uma imagem de identificação da conferência), rodapé e de

forma leve nos menus de navegação, construindo-se desta forma um template personalizado para o sítio "Conferência Experimental" (Figura 25: Escolha de Templates no Drupal e seu código CSSFigura 25).

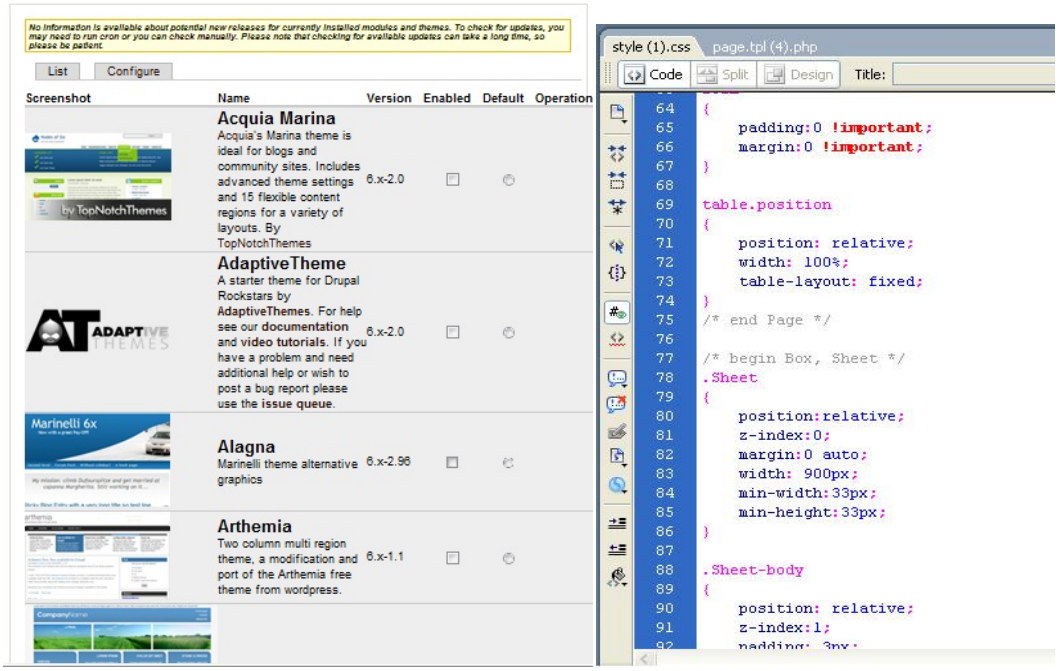


Figura 25: Escolha de Templates no Drupal e seu código CSS

Assim, e sem grande esforço, foi possível o desenvolvedor definir um template de apresentação personalizado a seu gosto para apresentação dos conteúdos criados antes.

"Marcação de Conferências na FCUL: Página Inicial

Em termos da aplicação "Marcação de Conferências na FCUL", desenvolvida sobre a plataforma DIF2.0, as primeiras funcionalidades a definir são as que estão englobadas na página inicial da aplicação, também vista como uma página de Login no serviço de marcação de conferências. Assim sendo, ao nível da página inicial procedeu-se a:

1. **Importação de informação presente no protótipo Drupal – "Conferência Experimental":**

Como parte da informação a mostrar no Sítio de Marcação de Conferências, foram importados conteúdos do sítio Drupal desenvolvido anteriormente, através de procedimentos AJAX e JAVASCRIPT já explicados neste documento ("Aspectos Técnicos da Integração Drupal – DIF2.0"), e foram englobados na Página Inicial como parte da mesma. Sempre que esses conteúdos são alterados no Drupal, as mudanças reflectem-se ao nível desta página, e por isso a mostragem dessa informação é considerada dinâmica. Na imagens seguinte (Figura 26) pode-se verificar que informação presente no protótipo desenvolvido em Drupal está também presente nesta página inicial.

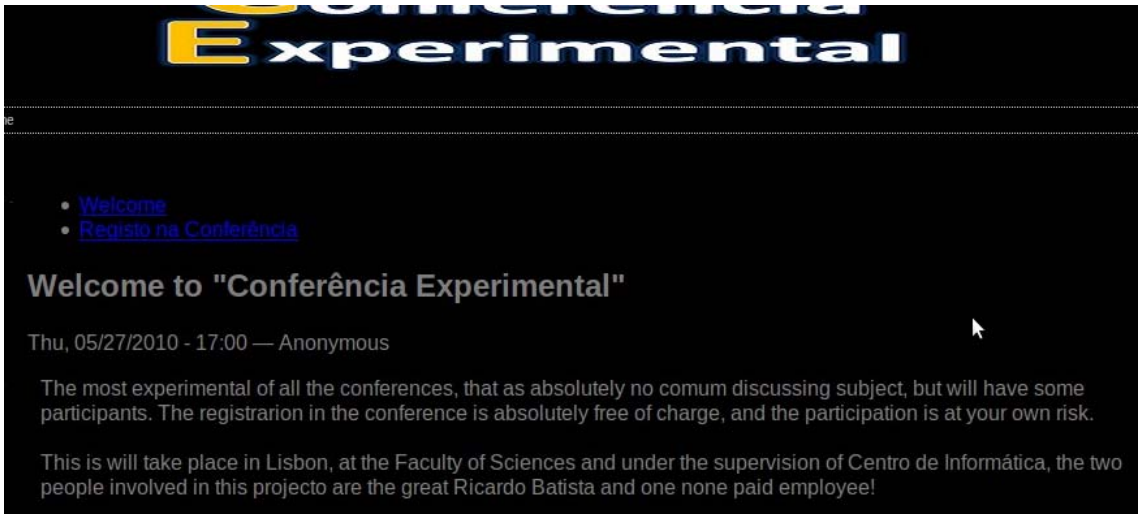


Figura 26: Imagem de texto importados do Drupal

Esta forma de importação de conteúdos foi a forma ideal encontrada para integração das duas ferramentas no desenvolvimento, e a sua implementação após algum período de aprendizagem e experimentação torna-se relativamente fácil para programadores com alguma experiência em linguagens de programação Web. Inclusivamente o desenvolvimento de métodos simplificados que recebem como parâmetro o URL do conteúdo a importar e respondem com variáveis com o conteúdo pretendido, facilita de sobremaneira a codificação deste tipo de tarefa.

2. Criação do mecanismo de Login no Serviço de marcação de conferências:

Ainda na página principal deste protótipo de serviço deu-se a implementação de um mecanismo de autentificação, que apresenta como principal destaque, o facto de fazer uso de um Web Service já implementado em Serviços em utilização na FCUL. No caso, o serviço de autentificação faz uso do serviço Web de autentificação no sítio da Intranet da FCUL, sítio esse. que representa uma plataforma de serviços para os alunos, e que por isso só garante acesso a utilizadores com credenciais - nome de utilizador e palavra-chave -, que os autentificam como Alunos da FCUL. Assim sendo, apenas é tornado possível o login nesta aplicação de “Marcação de Conferências” a alunos, professores e investigadores que já possuem credenciais para efectuar autentificação no portal em causa.

Resta referir que a integração deste Serviço de Autentificação nesta implementação é feita da mesma forma que é feita a importação de dados informativos do Drupal, fazendo uso de técnicas AJAX e JAVASCRIPT, para efectuar chamadas a URLS de Serviços Web e receber as respostas destes sobre a forma de variáveis que podem depois ser verificadas e utilizadas no âmbito da aplicação desenvolvida (Figura 27).

```
53
54      @OnSubmit("form")
55      public void submitForm(IDIFContext context){
56          if (!errors.hasErrors()){
57
58              try {
59                  URL url = new URL(
"http://webservices.fc.ul.pt/ci/UserAccountWS/Login.asmx/DoLogin?Username=" + username + "&Password=" +
password);
60
61                  URLConnection connection = url.openConnection();
62
63                  connection.setDoInput(true);
64                  connection.setUseCaches(false);
65
66                  DocumentBuilderFactory documentBuilderFactory =
67                      DocumentBuilderFactory.newInstance();
68
69                  documentBuilderFactory.setValidating(false);
70                  documentBuilderFactory.setFeature("http://xml.org/sax/features/namespaces", false);
71                  documentBuilderFactory.setFeature("http://xml.org/sax/features/validation", false);
72                  documentBuilderFactory.setFeature(
"http://apache.org/xml/features/nonvalidating/load-dtd-grammar", false);
73                  documentBuilderFactory.setFeature(
"http://apache.org/xml/features/nonvalidating/load-external-dtd", false);
74
```

Figura 27: Exemplo de código Java que implementada a chamada ao WebService de Autentificação

Prova-se assim com a implementação desta funcionalidade que é possível o uso de Serviços Web remotos e sua incorporação como parte das aplicações e serviços desenvolvidos na plataforma de desenvolvimento DIF2.0.

"Marcação de Conferências na FCUL": Registo de Conferências

A segunda página criada ao nível deste protótipo de serviço, é a página que permite o registo de conferências. Para isso é apresentado ao utilizador um formulário de preenchimento que permite criar conferências e alocar as mesmas a salas e a horas específicas. Para que isto fosse possível foi necessário:

1. Criação da Base de Dados para Marcação de Conferências:

Para que seja possível a marcação de conferências, é primeiro necessária a criação de uma base de dados onde armazenar toda a informação referente às mesmas. Assim sendo, o primeiro passo para implementar um sistema experimental de registo de conferências foi a criação de uma base de dados com tabelas para Conferências, Salas e Horas e que pudesse ser preenchida pela aplicação e respectivo formulário de registo de conferências (ponto seguinte deste "Registo de Conferências"). Assim sendo, iniciou-se a construção dessa base de dados em MySQL (ferramenta de gestão de bases de dados relacionais, mais utilizada ao nível de aplicações e serviços FCUL), para depois mais tarde se abandonar a mesma e reiniciar a construção em PostgreSQL, devido a problemas relacionados com a geração automática de classes sobre tabelas MySQL (descrito mais à frente ao pormenor na secção de problemas de implementação).

2. Configuração do Sistema para Geração de Classes de Acesso à Base de dados:

Após a criação das tabelas partiu-se para a configuração dos ficheiros da plataforma DIF2.0 que permitem a geração de classes de acesso à base de dados criada. Assim sendo foram realizados os seguintes passos:

2.1. Acrescentou-se a ferramenta usada para criação da base de dados (primeiramente MySQL, depois PostgreSQL), bem como a ferramenta usada para geração das classes de acesso a esta (hibernation), ao ficheiro de dependências do projecto (Figura 28):

```
<!-- dependencias do mysql ->
      <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
           </dependency>
<!-- mysql end -->
<!-- depedencias do hibernation -->
           <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate</artifactId>
                <version>3.2.3.ga</version>
           </dependency>
              <dependency>
                <groupId>c3p0</groupId>
                <artifactId>c3p0</artifactId>
                <version>0.9.1.2</version>
           </dependency>
<!-- hibernation end -->
```

Figura 28: Extracto do ficheiro de configuração de dependências do projecto

2.2. Criaram-se três ficheiros diferentes de configuração (Figura 29): O primeiro, a que é dado nome composto pelo nome da base de dados e uma extensão ".XML", contém as configurações de conexão á base de dados, como a sua localização, as credenciais de acesso e outros pormenores; O segundo, a que é dado o nome de "hibernate.reveng.xml", contém os passos necessários para a geração das classes de acesso às tabelas da base de dados, como a configuração dos nomes das tabelas e das suas chaves primárias; O terceiro, a que é dado o nome "orm.genetarion.xml", é responsável por nomear o serviço a criar para acesso à base de dados, e especifica quais as tabelas a que o mesmo vai aceder.

```
---------------------------------- exemplo de nomedabasededadados.xml--------------------------------------

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
```

```
        <property name="hibernate.connection.driver_class">org.gjt.mm.mysql.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost/diff</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">123</property>
        <property name="hibernate.default_schema">diff</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>
         <property
name="hibernate.current_session_context_class">org.hibernate.context.ThreadLocalSessionContext</property>
    </session-factory>
</hibernate-configuration>
------------------------------exemplo de hibernate.reveng.xml -------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC
 "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd" >
<hibernate-reverse-engineering>
    <type-mapping>
        <sql-type jdbc-type="DECIMAL" scale="0" hibernate-type="java.lang.Long" />
        <sql-type jdbc-type="DECIMAL" hibernate-type="java.lang.Double" />
        <sql-type jdbc-type="CHAR" hibernate-type="java.lang.Character" />
    </type-mapping>
    <table name="t_acl">
        <primary-key>
            <generator class="increment"></generator>
        </primary-key>
    </table>
    <table name="t_comentarios">
         <primary-key>
            <generator class="increment"></generator>
        </primary-key>
    </table>
</hibernate-reverse-engineering>
---------------------------------------------- exemplo de orm.genetarion.xml ------------------------------------
<model>
    <service name="Auth">
        <uses dao="Acl"/>
        <uses dao="Comentarios" />
    </service>
```

```
</model>
```

Figura 29: Extracto dos ficheiros criados para geração automática de classes de acesso à base de dados.

2.3. Acrescentou-se ao ficheiro de dependências a extensão responsável pela geração do código de classes de acesso à base de dados, e que permite especificar também a localização de criação dessas mesmas classes (Figura 30).

```
<plugin>
    <groupId>pt.digitalis</groupId>
    <artifactId>maven-orm-generator-plugin</artifactId>
    <configuration>
      <dataSourceName>nomedabasededados</dataSourceName>
      <packageName>pt.dif.model</packageName>
    </configuration>
    <executions>
      <execution>
        <phase>generate-sources</phase>
        <goals>
          <goal>run</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
```

Figura 30: Extracto da importação da extensão Maven responsável pela geração automática de classes

2.4. Executar a geração automática de classes através da execução do comando Maven "mvn generate-sources", e retirar do ficheiro de dependências a extensão adicionada em 2.3.

3. Criação do Formulário de Registo de Conferências:

A criação de um formulário (Figura 33) e respectivos campos para preenchimento de tabelas numa base de dados criada previamente, faz-se na plataforma DIF2.0 com maior facilidade do que numa típica programação em PHP. Como já foi descrito nos passos anteriores, a DIF2.0 possui um mecanismo de geração de classes que fazem a comunicação entre a base de dados e as aplicações desenvolvidas na plataforma, simplificando a forma como são escritas e devolvidas informações de bases de dados. Aliando este facto ao pormenor de utilização de pequenas anotações sobre o código de uma aplicação, que permitem definir parâmetros e formas de validação dos mesmos, conclui-se que o desenvolvedor tem este tipo de tarefa grandemente facilitada. Assim, para construção deste formulário foi a executada a seguinte sequência de acções:

3.1. Criação das variáveis desejadas como campos de formulário ao nível da classe java e escrita de anotações indicando restrições sobre dados inseridos nestes parâmetros, ambas através da anotação "@param" e seus atributos (Figura 31).

```
@StageDefinition(name = "Home", service = "simpleformserv", overrideDefault = "difhomestage")
@View(target = "home.jsp")
@Callback
public class SimpleFormHome {


    @Inject
    IAuthService iAuthService;

    @InjectParameterErrors
    ParameterErrors errors;

    String stage = "";

    @Parameter(linkToForm= "form", constraints= "required")
    String username;

    @Parameter(linkToForm="form", constraints= "required")
    String password;


    @OnSubmit("form")
```

Figura 31: Exemplo de código que define os parâmetros de um formulário

3.2. Ligação das variáveis a acções a realizar sobre a base de dados fazendo para isso, uso dos métodos gerados para inserção de informação na base de dados criada anteriormente (Figura 32).

```
    @OnSubmit("form")
    public void submitForm(IDIFContext context) throws ParameterException{

        if (iAuthService.getConferenciasDataSet().query().in("nome", nome).count() > 0)
            errors.addParameterError("nome", new ParameterError("Conferencia duplicada",
                    ParameterErrorType.RULE));

        if(iAuthService.getSalasDataSet().query().in("sala", sala).count() > 0)
            for(Salas s: iAuthService.getSalasDataSet().query().in("sala", sala).asList())
                if(s.getDataInicio().before(datafim) && s.getDataFim().before(datainicio))
                    errors.addParameterError("sala", new ParameterError("Sala qualquer coisa",
                            ParameterErrorType.RULE));

        if(!errors.hasErrors()){
            iAuthService.getConferenciasDataSet().insert(new Conferencias(nome));
            iAuthService.getSalasDataSet().insert(new Salas(
                    iAuthService.getConferenciasDataSet().get(iAuthService.getConferenciasDataSet().query().
in("nome", nome).singleValue().getId().toString()),sala, datainicio, datafim));
        }
    }
```

Figura 32: Exemplo de código que mostra a invocação de métodos de inserção de conteúdos numa base de dados

Prova--se assim a forma como, de forma simplificada, são criados formulários ao nível de uma aplicação desenvolvida na DIF2.0 e como são automaticamente gerados os métodos responsáveis pela

manipulação da informação ao nível da base de dados. Interessa referir ainda que os passos 1 e 2 descritos são apenas realizados uma vez para cada base de dados criada num projecto.

Figura 33: Imagem da Página de Registo de Conferências

"Marcação de Conferências na FCUL": Página de Sucesso

Após criação de um formulário que permite preencher as tabelas da base de dados criada, e por isso permite criar e registar conferências ao nível de uma aplicação, foi desenvolvida uma página informativa, usualmente denominada por "Página de Sucesso", que faz a amostragem de conferências já registadas, servindo-se para isso de uma tabela que mostra os diversos campos informativos de cada conferência. Para isto, foi apenas necessário definir um campo DIF2.0 (denominada "grid") que define uma tabela e fazer a sua ligação com os campos da base de dados que se pretendia mostrar. Essa ligação foi feita novamente, através da utilização dos métodos auto-gerados pelo sistema para amostragem de campos presentes na base de dados, percorrendo todo o conteúdo da tabela "Conferencia" e devolução dos campos relacionados presentes nas restantes tabelas com ligação directa a esta (Figura 34).

```
@OnAJAX("conferenciasfill")
public IJSONResponse getMovies() {
    String[] fields = {"id","nome"};
    return new JSONResponseDataSetGrid(iAuthService.getConferenciasDataSet(), fields);
}
```

Figura 34: Código que implementa uma grid com campos da base de dados

De referir por fim, que para que um utilizador seja encaminhado para esta página é naturalmente obrigatório que sido feita o correcto registo de uma nova conferência (daí a denominação de "Página de Sucesso").

"Marcação de Conferências na FCUL": Alteração de Conferências

À semelhança do que foi feito nos dois casos anteriores, a criação de uma página para alteração de informação relacionada com conferências já registadas, é feita através da utilização de anotações que definem parâmetros de um formulário para alteração dos vários atributos de uma conferência, e através da utilização de métodos que manipulam informação presente na base de dados, e que permitem, tanto fazer corresponder a informação presente inicialmente no registo de uma conferência aos diferentes campos do formulário, como permite que alterações sobre estes campos sejam depois reflectidas ao nível das tabelas na base de dados.

Com a implementação destes três casos de manipulação de informação presente em bases de dados através de uma aplicação Web desenvolvida na DIF2.0, prova-se a fácil utilização deste tipo de conteúdos, tal como se dá uma ideia mais exacta do tipo de tarefas necessárias de executar para implementar este tipo de funcionalidades.

Após descrição das diferentes funcionalidades (ou atributos) implementadas ao nível deste protótipo de serviço Web, interessa agora fazer um levantamento dos problemas verificados ao nível desta plataforma de desenvolvimento, bem como dar uma ideia do esforço relacionado com a implementação das funcionalidades descritas.

Estudo de Problemas e Esforço na Implementação do Caso de Estudo

Após toda a descrição da implementação do caso de estudo, interessa especificar todo o tipo de problemas encontrados durante a realização do caso de estudo, bem como de alguma forma medir o esforço verificado no seu desenvolvimento. Só com base nessas especificações, é possível ter uma ideia real das limitações neste tipo de desenvolvimento conjunto que utiliza duas ferramentas distintas para um desenvolvimento convergente e posteriormente compará-lo com outras soluções de desenvolvimento que utilizam uma única ferramenta para obter os mesmos resultados.

Assim, passo a identificar em primeiro lugar, pormenores relacionados com o esforço de desenvolvimento, para de seguida listar os problemas encontrados na implementação:

Pormenores de Esforço de Desenvolvimento

Para implementação deste caso de estudo foi necessário:

- O uso do gestor de conteúdos Drupal e da plataforma rápida de desenvolvimento DIF2.0, tendo sido desenvolvidos os protótipos, por mim, José Coelho e por Ricardo Batista (bolseiro no CI-FCUL) num período superior a uma semana no Centro de Informática da FCUL.

- O protótipo desenvolvido em Drupal, foi realizado em menos de duas horas, mesmo tendo em conta o facto de não haver na altura um total conhecimento das formas de utilização da ferramenta.

- O protótipo desenvolvido na plataforma DIF2.0, foi implementado num período aproximado de uma semana e meia, sendo que a maior parte desse tempo, esteve relacionado com a aprendizagem de pormenores da plataforma, e com a resolução de problemas não calculados a início na implementação.

- A instalação e configuração inicial da ferramenta Drupal foram feitas de forma muito simples e intuitiva, havendo sempre documentos de suporte para esclarecimento de possíveis dúvidas na sua instalação e configuração.

- A instalação e configuração inicial da plataforma DIF2.0 foi efectuada com alguma dificuldade inicial, tendo sido perdido algum tempo na percepção de alguns pormenores técnicos e requisitos necessários à sua configuração e início de utilização, e tendo sido necessário o envio de dúvidas para a equipa de desenvolvimento responsável pela plataforma.

- No que respeita ao esclarecimento de dúvidas de configuração da plataforma de desenvolvimento, não tendo sido obtidas respostas no imediato (próprio dia), foi-se desta forma prolongando o tempo inicial de configuração da plataforma DIF2.0, atrasando a implementação do caso de estudo.

Problemas Verificados no Desenvolvimento:

Após um período inicial de experimentação e utilização das componentes de desenvolvimento que a plataforma oferece, fica a ideia de que esta não está ainda muito desenvolvida, não se traduzindo em tão grande número e diversidade de módulos como seria de esperar (e como ficara a ideia após a formação inicial) apresentando-se bastante incompleta a vários níveis. Níveis esses que passo a especificar:

- Módulos relacionados com ligações a bases de dados e com manipulação de informação derivada de fontes externas apresentaram-se com muitos erros e ainda numa fase inicial de desenvolvimento. Isto verifica-se uma vez que na tentativa de implementação de um caso de teste com manipulação de informação presente numa base de dados MySQL, os métodos geradores das interrogações sobre o modelo de dados, resultaram em erros relacionados com o facto do código SQL gerado automaticamente pela plataforma estar mal definido no que à lógica da aplicação diz respeito. Devido a este facto, a implementação de um protótipo na DIF2.0 teve de ser reconfigurada para utilização de uma base de dados desenvolvida em PostgreSQL, abandonando a ferramenta MySQL correntemente utilizada na maior parte dos serviços oferecidos pelo CI-FCUL.

- Verificou-se alguma dificuldade na identificação geral do funcionamento dos métodos fornecidos pela plataforma, bem como a inexistência de suporte básico sobre os mesmos. Mais especificamente, exemplos de funcionamento ou documentação esclarecedora dos métodos e das classes disponibilizadas existem em quantidade muito reduzida e apenas no sítio oficial da plataforma, o que leva o programador estreante a perder-se na implementação de uma funcionalidade, traduzindo a

implementação de funcionalidades numa experiência baseada em tentativa-erro na utilização das várias classes.

- Na implementação de formulários que pretendem tratar informação dinâmica presente em bases de dados (no caso apenas em bases de dados PostgreSQL e não MySQL, como já foi referido), a configuração de restrições sobre os campos do formulário nem sempre resulta da forma pretendida. Como exemplo disto, temos o facto de, aquando da implementação de restrições de obrigação de preenchimento de campos, nem sempre ser colocado um "*" em campos especificados como obrigatórios e não ser tão pouco verificada essa restrição. Verificou-se também que este tipo de comportamento está dependente do tipo de dados que se configura para inserção nos campos (no caso, o tipo de dados Date).
- Derivado do mecanismo de geração automática de classes para acesso a conteúdos em base de dados, é necessário haver alguns cuidados aquando da criação das tabelas que irão conter os conteúdos, uma vez que no caso de existência de caracteres como "_", este mecanismos não é correctamente executado, resultando em erros de compilação. Verificou-se também que a nomeação das tabelas deve ser feita, sempre utilizando uma letra maiúscula no início de cada nome. Como exemplo disto, é de notar que foi necessário converter alguns nomes na base de dados inicialmente criada para que: a tabela "T_conferencias" passasse a "Conferencias" e os atributos CD_QUALQUER, DT_INICIO e DS_ALGUMA_COISA passassem a CDQualquer, DateInicio e DescAlgumaCoisa.
- Aquando da importação das bibliotecas de funcionalidades na ferramenta Eclipse para uso da plataforma DIF2.0, o comportamento é bastante instável, resultando muitas vezes no não reconhecimento de vários métodos e na falha da importação dos mesmos, tendo que ser repetida uma sequência de acções para que as bibliotecas acabem por ser novamente reconhecidas para a plataforma, e para que as funcionalidades possam ser novamente utilizadas por via programática.
- Outro facto claramente negativo na utilização da plataforma de desenvolvimento em causa, relaciona-se com o facto de os tempos de compilação de um projecto serem muito longos e resultarem num tempo de espera substancial sempre que se protagoniza alguma alteração no código do projecto. Assim sendo, o tempo para verificação de cada funcionalidade que é desenvolvida é demasiado longo, tornando o processo de criação demasiado lento para uma plataforma de rápido desenvolvimento de aplicações.
- Como último ponto, destaca-se ainda o facto do trabalho e desenvolvimento sobre a plataforma, depender de uma ligação constante dos computadores que a correm à Internet. Pode-se mesmo dizer que a implementação na DIF2.0, não é possível off-line, devido à grande quantidade de componentes que dela dependem e ao constante descarregamento de informação e módulos a cada compilação do código de um projecto.

Conclusões sobre Caso de Estudo

Conclui-se da realização deste caso de estudo e de todos os pormenores da sua realização, bem como dos problemas levantados e do esforço de desenvolvimento verificado, que o desenvolvimento

conjunto de aplicações com base no gestor de conteúdo Drupal e na plataforma de desenvolvimento DIF2.0, é uma possibilidade que deve ser tomada em conta para o processo de desenvolvimento futuro a implementar no Centro de Informática da FCUL. No entanto, e devido à grande quantidade de problemas verificados, conclui-se também que só com um maior uso e experimentação profunda da plataforma em causa, se poderá ter uma ideia mais exacta de tudo aquilo que será possível de implementar com recurso a esta, e de criar automatismos e hábitos na sua utilização, que permitam acelerar a implementação de projectos na mesma.

Sobre a integração das duas ferramentas no desenvolvimento, pode-se dizer que embora o uso simultâneo de ambas permita simplificar a implementação e gestão de conteúdos mais simples, e que um maior número de utilizadores possa participar em processos de desenvolvimento, em contrapartida, a necessidade de uma integração constante de vários elementos através de código, obriga a uma grande dependência de programadores mais experientes na gestão de projectos mais complexos. Pode-se dizer ainda, que o facto de projectos desenvolvidos na DIF2.0 integrarem informação presente em páginas Drupal, obriga a que a alteração exclusiva do conteúdo importado, sem alteração na fonte, seja tarefa unicamente ao acesso de programadores experientes e não dos típicos utilizadores sem conhecimentos de programação que originalmente criaram o conteúdo no Drupal. Assim sendo, pode-se concluir que a forma de integração e uso de duas ferramentas para um desenvolvimento conjunto, funciona, e possibilita um desenvolvimento integrado, mas que no entanto não simplifica assim tanto o acto de gestão da informação, como se pretenderia no início deste projecto, podendo haver alternativas mais simples de conseguir o mesmo desenvolvimento.

A juntar a tudo isto, ressalta ainda de toda a utilização da plataforma de desenvolvimento DIF2.0, que existe uma grande falta de documentação e de distribuição de conhecimento, no que diz respeito à mesma, e que a aquisição desse conhecimento pode ser uma tarefa complexa, e apenas possível com a colaboração da própria equipa que desenvolveu o produto. Existem ao nível desta ferramenta muitas falhas que não eram de prever a início, e a sua resolução revela-se essencial para um bom desenvolvimento na plataforma, dependendo mesmo disso a adopção real da DIF2.0 no CI-FCUL.

5.2 Desenvolvimento Unicamente em Drupal

Durante todo este projecto foi abordada uma solução de desenvolvimento tecnológico para o CI baseado em dois tipos de ferramentas, um gestor de conteúdos para gestão e carregamento de dados estáticos bem como para a construção de sítios Web simples, e uma plataforma de desenvolvimento mais complexa para implementação de verdadeiros serviços Web que manipulam dinamicamente informação, lendo e actualizando dados presentes em bases de dados ou em outras fontes externas. No entanto, e como resultado da utilização do Drupal para construção de pequenos sítios Web de exemplo, e real arranque no CI de projectos de implementação de verdadeiros sítios Web usando esta ferramenta, levantou-se uma possibilidade de desenvolvimento que a início não tinha sido pensada, o uso do Drupal como ferramenta única de desenvolvimento de serviços e consequente abandono da plataforma de desenvolvimento DIF2.0.

Esta hipótese surge após verificação das capacidades do Drupal na construção de sítios Web, e após percepção da capacidade desta ferramenta para construção de sítios Web com informação dinâmica, através da simples instalação de módulos que permitem manipular dados em bases de dados MySQL (tipo de bases de dados mais usada em todos os serviços implementados na FCUL), bem como de módulos que fornecem a um programador uma grande quantidade de funcionalidades, tipicamente presentes em verdadeiros Serviços construídos por plataformas de desenvolvimento.

Para que se possa considerar a hipótese de forma real é no entanto necessário uma análise atenta das possibilidades e limitações reais do Drupal como ferramenta única de desenvolvimento, e a comparação entre essa hipótese e a hipótese de desenvolvimento conjunto com a plataforma de desenvolvimento DIF2.0. Neste capítulo é feito todo esse estudo pormenorizado, chegando-se a uma conclusão final sobre a melhor opção de desenvolvimento para o CI-FCUL.

5.2.1 Drupal como Ferramenta de Desenvolvimento para o CI-FCUL

Ao considerar-se uma opção de desenvolvimento, baseada exclusivamente no que inicialmente seria apenas uma ferramenta de gestão de conteúdos, tem-se a declarada intenção de simplificar todo o processo de desenvolvimento ao mesmo tempo que se estende a um maior número de desenvolvedores (principalmente aos que têm relativamente pouca experiência em programação) a implementação de serviços Web que a início apenas estavam ao alcance de alguns desenvolvedores. O desenvolvimento de todo o tipo de conteúdos Web numa única ferramenta, permite também evitar redundâncias na informação, e concentrar esforços no simples desenvolvimento de componentes ao invés de o dispersar por tarefas de integração das mesmas por várias ferramentas conjuntas de desenvolvimento.

Existem vários pormenores relacionados com o desenvolvimento unicamente no Drupal, que devem ser considerados e estudados, nos quais se inclui o estudo dos diversos tipos de conteúdos que o CMS permite desenvolver, a forma como diferentes sítios Web devem ser apresentados tendo em conta a possibilidade de existência de variantes de templates de apresentação no Drupal e a forma como o

desenvolvimento deve ser feito tendo em conta os vários desenvolvedores presentes no CI-FCUL que poderão ter competências para o levar a cabo. Todos estes pormenores são discutidos de seguida:

Conteúdos Drupal

Todo o desenvolvimento de conteúdos no Drupal se processa de forma relativamente simples, e sobretudo de uma forma ao alcance de quase qualquer utilizador da ferramenta, quer este tenha conhecimentos profundos de programação ou não. Isto acontece, uma vez que toda a implementação de funcionalidades no Drupal se baseia no uso de módulos que simplificam a criação de vários tipos de conteúdos (Figura 35 e Figura 36), permitindo que quase tudo seja efectuado de forma análoga à navegação por um sítio através da selecção básica de opções e do simples preenchimento de campos. A existência de módulos para quase todos os conteúdos imagináveis e o simples funcionamento de todos eles, fazem do Drupal uma ferramenta de desenvolvimento muito poderosa, possibilitando a criação de Serviços e Aplicações outrora complexas, de forma relativamente simples e intuitiva.

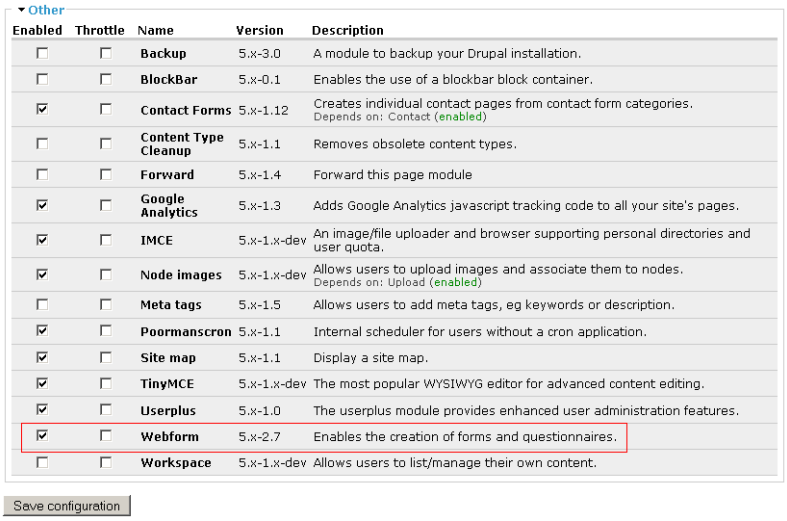


Figura 35: Instalação de Módulos para Uso no Drupal

Uma das características mais marcantes do desenvolvimento baseado apenas em Drupal, é a forma como conteúdos estáticos e dinâmicos são vistos do ponto de vista do desenvolvimento, como tendo o mesmo tipo de dificuldade de implementação. Operação como a típica actualização de páginas de notícias, a construção de painéis gerais que permitem agregar vários tipos de informação, a criação de formulários que manipulam informação presente em bases de dados e a integração com fontes de informação externas ao Drupal, são todas vistas como tendo graus de complexidade de implementação idênticos e como podendo ser feitas por quase qualquer tipo de desenvolvedor.

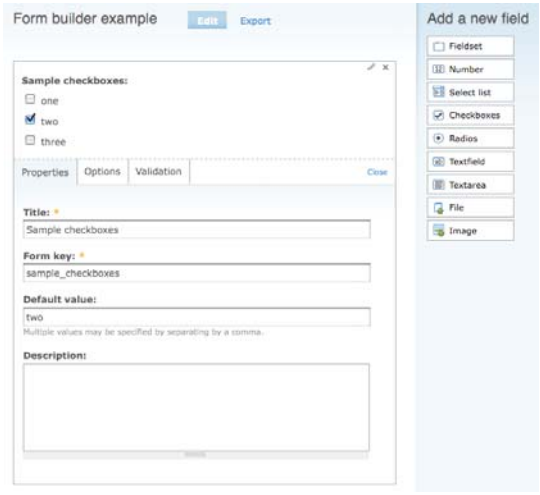


Figura 36: Criação de um formulário usando um módulo para esse efeito

Outro dos pontos mais interessantes desta ferramenta de criação e gestão de conteúdos, é o facto de continuar a permitir a programação de conteúdos em PHP e HTML por parte de programadores e desenvolvedores experientes. Isto é, para além de haver um grande número de possibilidades de criação de conteúdos através da instalação e uso de módulos de funcionalidades, o Drupal também permite o desenvolvimento e programação de módulos que implementam funcionalidades, por parte dos seus utilizadores, possibilitando que estes façam uso dos seus conhecimentos programáticos ao nível do desenvolvimento de aplicações e serviços, e permitindo que conhecimentos ao nível de codificação em linguagens Web sejam aproveitados conjuntamente com a simplicidade de criação de conteúdos num nível mais alto de abstracção de código.

Uma maior pormenorização do desenvolvimento de conteúdos em Drupal será ainda dada mais à frente aquando da apresentação dos casos de estudo de sites desenvolvidos em Drupal para o CI-FCUL.

Templates de Apresentação no Drupal

Outra vantagem do desenvolvimento único em Drupal relaciona-se com a criação e gestão dos templates e dos aspectos de apresentação nos diferentes serviços e aplicações. Uma vez que não há a preocupação em construir duas vezes o mesmo template de apresentação em linguagens de programação diferentes para plataformas de desenvolvimento diferentes, haverá acima de tudo, a preocupação de criação de diferentes tipos de apresentações, ou de templates de apresentação com estruturas idênticas mas com pequenas variantes que permitam identificar do ponto de vista visual, diferenças e semelhanças ao nível dos serviços e das aplicações desenvolvidas. Assim sendo:

Uniformização dos Templates de Apresentação

A uniformização dos templates de apresentação ao nível do desenvolvimento em Drupal, confere-se ao desenvolvimento de formas relacionadas de apresentação dos conteúdos com vista a que todos os produtos do desenvolvimento (aplicações e serviços) se relacionem visualmente e possam ser

identificados univocamente como pertencentes à mesma instituição. Isto é, interessará ao Centro de Informática e à própria FCUL, que serviços a si pertencentes sejam de alguma forma identificados de forma semelhante, e que por isso a sua estrutura de apresentação visual, apresente semelhanças que os permitam identificar como pertences à mesma instituição ou ao mesmo grupo de desenvolvimento. Tal como interessará que aplicações e serviços desenvolvidos com o mesmo fim ou para o mesmo tipo de utilizadores (exemplo: construção de sítios de departamentos, construção de páginas pessoais de professores e construção de sistemas de gestão de contas ou de sistemas de inscrições) tenham templates de apresentação muito idênticos como forma de manter a eficiência na sua utilização e de habituar o utilizador a efectuar sempre o mesmo tipo de operações sobre o mesmo tipo de aplicações. Este tipo de uniformização nos templates de apresentação é facilmente atingido no Drupal, através da construção de determinados tipos de templates e sua consequente utilização por sítios e aplicações que partilham as mesmas características, para que aplicações com os mesmos fins usem o mesmo template de apresentação.

Templates de Apresentação Não Uniformizados

Tal como é necessária a utilização de templates de apresentação iguais em aplicações que servem os mesmos propósitos ou que se relacionam no tipo de serviço que oferecem ou no tipo de sítio que constituem, também será necessária alguma variância ao nível dos templates de apresentação para que não fique a sensação que todos os serviços ou aplicações são iguais, ou para que haja confusão na identificação de serviços semelhantes mas que servem identidades diferentes. Assim sendo, sabe-se em primeiro lugar, que será sempre necessário o desenvolvimento de diferentes templates de apresentação, consoante o tipo de aplicação a desenvolver, isto é, serviços de inscrições serão sempre apresentados de forma diferente do que sítios informativos pessoais ou sítios de apresentação de departamentos (Figura 37), e por isso deverão também fazer uso de diferentes templates de apresentação, e ser nesta medida não uniformes.

Figura 37: Exemplos de sítios com diferentes templates - Um sítio de uma conferência e um portal de gestão interna

Para além de ser necessário criar diferentes templates para serviços, sítios e aplicações com propósitos diferentes, também convirá possibilitar alguma mudança visual no que a sítios com propósitos idênticos diz respeito, para que possa haver alguma personalização no aspecto que permita separar e evitar confusões entre sítios que embora sejam do mesmo género, não servem o mesmo grupo de utilizadores. Este tipo de personalização pode ser feito no Drupal através da construção de templates idênticos mas com pormenores diferentes no que diz respeito a cores gerais, a imagens de cabeçalhos, a cores de menus, a tipos de letras, etc., mantendo a mesma estrutura visual e o mesmo arranjo nos blocos informativos (Figura 38).

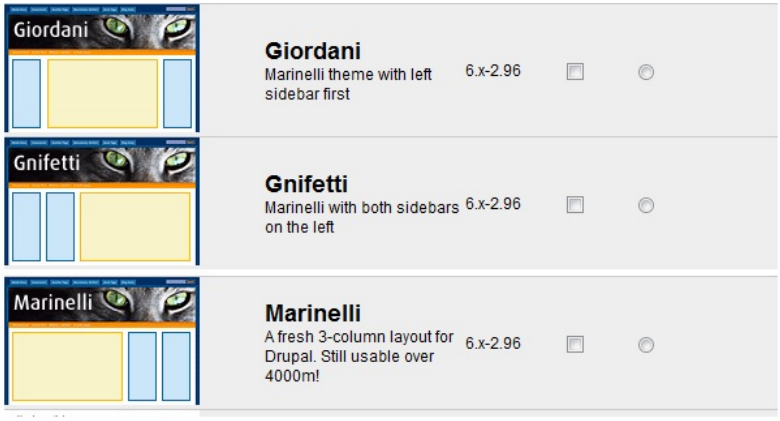


Figura 38: Exemplo de templates com pequenas alterações na sua estrutura

Gestão de Utilizadores no Drupal

Sendo um facto que o desenvolvimento utilizando unicamente o Drupal, permite englobar um maior número de técnicos do CI-FCUL na directa implementação de funcionalidades para serviços e aplicações Web, também é um facto que deve haver alguma separação no tipo de tarefas de implementação que são realizadas, atribuindo tarefas de grau de responsabilidade mais elevado a desenvolvedores com um maior nível de responsabilidade e maior experiência no desenvolvimento deste tipo de aplicações. Se isto é verdade para tarefas relacionadas com a implementação de funcionalidades em projectos, também o é no que diz respeito a tarefas de configuração da ferramenta, à instalação de módulos de funcionalidades ou à gestão e criação dos próprios utilizadores. No entanto, toda essa separação de tarefas e de responsabilidades é assunto relacionado apenas com políticas internas da gestão da Instituição, não sendo a metodologia ou o processo de desenvolvimento que os definem de forma padronizada, e por isso o único ponto certo é o de que o Drupal, permite a quase todos os utilizadores a realização de quase todos os tipos de tarefas, quer o utilizador tenha conhecimentos programáticos, quer não os possua. Tudo depende de um curto período de aprendizagem sobre o funcionamento dos diferentes módulos que permitem a implementação das diversas funcionalidades. Assim sendo, na tabela seguinte são divididos os diferentes tipos de tarefas no Drupal pelos diferentes tipos de desenvolvedores capazes de as levar a cabo:

Tarefa no Drupal	Tipo de Utilizador:		
	Sem Conhecimentos Programáticos	Com Conhecimentos Programáticos Básicos	Programador
Desenvolvimento de conteúdos estáticos	Sim	Sim	Sim
Desenvolvimento de conteúdos dinâmicos	Sim	Sim	Sim
Gestão de conteúdos estáticos	Sim	Sim	Sim
Gestão de conteúdos dinâmicos	Sim	Sim	Sim
Instalação de Módulos de Funcionalidades	Sim	Sim	Sim
Implementação de Templates de Apresentação	Não	Sim	Sim
Gestão e Instalação de Templates de Apresentação	Sim	Sim	Sim
Criação e Gestão de Utilizadores	Sim	Sim	Sim
Criação de Endereços/Espaços para Criação de Aplicações Web	Sim	Sim	Sim
Desenvolvimento de novos Módulos de Funcionalidades	Não	Não	Sim

Tabela 8: Tarefas Drupal e Tipo de Utilizadores que as realizam

Como se pode ver pela primeira coluna, quase qualquer tipo de utilizador consegue realizar qualquer tipo de tarefa, a não ser obviamente as tarefas relacionadas exclusivamente com programação e que não podem de forma alguma ser "atalhadas" via módulos de funcionalidades do Drupal.

Para um melhor entendimento de todas as capacidades e limitações do Drupal como ferramenta de desenvolvimento para o CI-FCUL, segue-se um caso de estudo de diferentes sítios Web desenvolvidos em Drupal para a FCUL, tanto nos termos deste projecto, como no âmbito de projectos anteriores e que são agora revisitados e estudados como oportunamente no âmbito do projecto.

5.2.2 Caso de Estudo: Sítios em Drupal para a FCUL

Ao contrário do caso de estudo feito para a Plataforma de Desenvolvimento DIF2.0, para o Drupal foi feito outro tipo de estudo também baseado na implementação de um protótipo de sítio Web para o CI, mas essencialmente no estudo de um projecto já realizado em Drupal, no passado, por um membro do CI-FCUL, para a própria Instituição. O estudo da forma como foram implementados os diversos tipos de funcionalidades, bem como o estudo do esforço de desenvolvimento destas aplicações, são essenciais para perceber as capacidades e limitações da ferramenta.

Funcionalidades Implementadas

No total, este caso de estudo compreende um Sítio Web desenvolvido sobre Drupal e já concluído no passado (há cerca de mais de um ano), e um protótipo simples desenvolvido no âmbito deste projecto, e sobre o qual foi apenas experimentada a implementação de funcionalidades básicas do Drupal, como forma de perceber o esforço necessário na implementação de um projecto utilizando esta ferramenta. São estes dois exemplos que passo a descrever de seguida, primeiro de uma forma geral para depois fazer referências às funcionalidades desenvolvidas sobre estes, para por fim fazer referência a algumas funcionalidades não implementadas mas que se revelam possíveis de implementação no Drupal.

Visão Geral das Aplicações

Assim sendo, e partindo primeiro para o exemplo da aplicação já concluída, e depois para o protótipo experimentado, destaco os sítios Web localizados em http://dbv.fc.ul.pt (Figura 39), e internamente em http://ze.ci.fc.ul.pt (Figura 40).

O primeiro caso, é um sítio Web desenvolvido para o Departamento de Biologia Vegetal (DBV) da FCUL (Figura 39) que compreende um grande conjunto de páginas com conteúdo estático e informativo, donde de se destaca a página de "Apresentação", vários conjuntos de páginas que fornecem informação sobre a "Organização", o "Ensino", a "Investigação", e os "Recursos Humanos" do Departamento, bem como ainda um conjunto de páginas que fornece informação aos Alunos ("Alunos") e outra pequena página de "Contactos". Pelo meio de todas estas páginas, destacam-se as funcionalidades implementadas pelas páginas referentes aos "Recursos Humanos", que permitem que informação referente a Docentes e Investigadores seja carregada para o Sítio directamente de uma Active Directory ("directório activo"), bem como as páginas que fornecem informação a Alunos ("Informações Úteis", "Horários", "Planos de Estudo" e "Disciplinas" dentro do menu "Alunos") que possibilitam o descarregamento de ficheiros com informação útil para os utilizadores. Para além de todas estas páginas, o sítio fornece ainda um sistema de pesquisa, colocado no topo da barra direita de navegação, uma pequena funcionalidade de calendário para consulta de eventos, um sistema de autentificação para alunos que fazem parte do Departamento e um formulário (na página "contactos") para que qualquer utilizador possa enviar mensagens de correio electrónico para o Departamento.

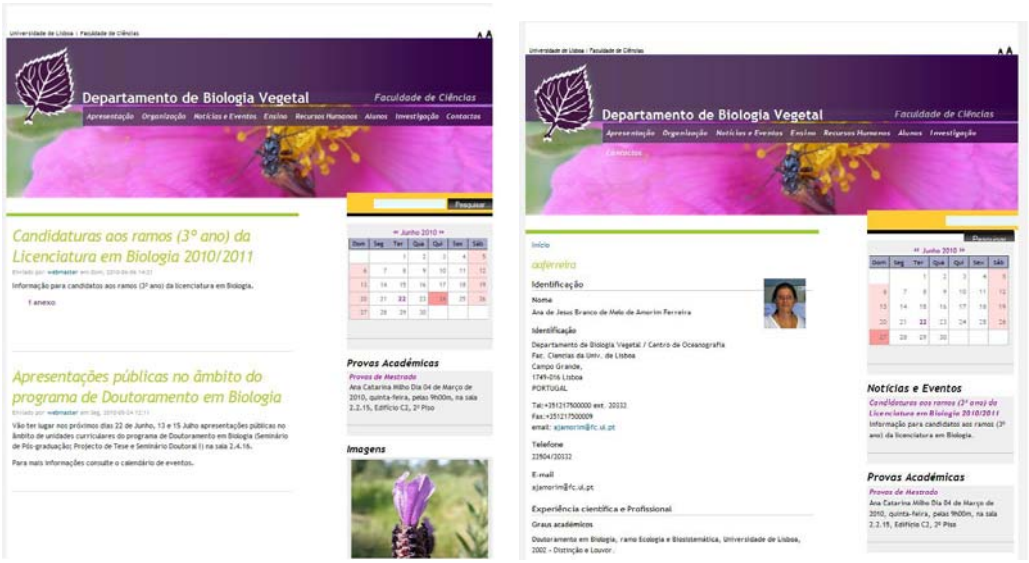


Figura 39: Imagens do Sítio Drupal desenvolvido para o DBV

O segundo caso de desenvolvimento integrante deste caso de estudo, é um protótipo de um Sítio Web que pretende ser um exemplo de importação do sítio oficial do Centro de Informática (http://ci.fc.ul.pt) para Drupal (Figura 40). Este protótipo é constituído por um conjunto de páginas de conteúdo estático de informação, um conjunto de menus que permite a navegação entre páginas e principalmente por um template que tem por objectivo fazer uma aproximação à forma de apresentação do sítio do oficial do CI desenvolvido em HTML e PHP.

Figura 40: Imagem do Protótipo Drupal do Sítio Web do CI

Estudo das Funcionalidades Implementadas

Verificando caso a caso as funcionalidades desenvolvidas em cada um dos protótipos de aplicação desenvolvidos, pode-se ter uma ideia mais fiel do que é de facto o desenvolvimento de aplicações Web no Drupal. No entanto, e ao contrário do que acontece no caso de estudo do desenvolvimento conjunto apresentado anteriormente, a análise do desenvolvimento é muitas vezes feito - no caso do primeiro protótipo desenvolvido - de forma superficial e não do ponto de vista do desenvolvedor, uma vez que se baseia em relatos de desenvolvimento de quem foi responsável por este, e não directamente na minha experiência de utilização do Drupal nos casos em causa. Assim sendo:

"Sítio DBV" e "Sítio CI-FCUL": Páginas Informativas

Tal como aconteceu no caso de estudo apresentado anteriormente, a implementação de páginas informativas (Figura 41), e portanto de conteúdo estático, no Drupal é muito simples e baseado no uso de editores simplificados de texto (ver criação de páginas informativas no caso de estudo referente ao desenvolvimento conjunto de aplicações). Com a instalação simples de módulos que permitem essa edição simplificada, qualquer utilizador é capaz de criar e gerir conteúdo, como se tivesse a utilizar um típico programa de edição de texto.

Para além de um módulo para edição simplificada de texto, existem outros módulos que têm por objectivo permitir a elaboração de conteúdo composto no Drupal, isto é, conteúdo que integra vários tipos de outros conteúdos, um exemplo disto é página de notícias implementada neste sítio, que faz uso de um módulo de Taxonomia (Taxonomy - http://drupal.org/project/Taxonomy_Pages) que permite agrupar informação por pontos de interesse em comum, ou por referências ou assuntos relacionados e depois visualizar a mesma sobre a forma de várias páginas encadeadas numa lista.

Figura 41: Exemplo de uma página informativa no Drupal (Protótipo do Sítio do CI)

Uma página de notícias pode também fazer uso de um módulo de Livro (Book - http://drupal.org/handbook/modules/book) que permite da mesma forma agrupar um conjunto de conteúdos do mesmo género num mesmo "livro" que pode depois ser visualizado "página a página". Ou ainda funcionar sobre a forma de um painel (Panels - http://drupal.org/project/panels) que agrega vários tipos de informações por várias colunas, servindo-se do módulo com o mesmo nome.

Ainda no que diz respeito a módulos Drupal que permitam a integração de vários tipos de conteúdos num só, convém ainda referir o módulo denominado "Vista" (Views - http://drupal.org/project/views) (Figura 42) que tem como principal função a criação de blocos de informação, de páginas ou de menus, agregando vários tipos de conteúdos numa única página ou pedaço de informação, sendo ainda possível definir parâmetros para esse agrupamento, que vão desde o factor de ordenação (quais os conteúdos que são mostrados em primeiro lugar), ao factor de apresentação (quantos conteúdos são mostrados de cada vez) ou ainda a factores de filtro que permitem excluir determinado tipo de informação presente em conteúdos do mesmo género.

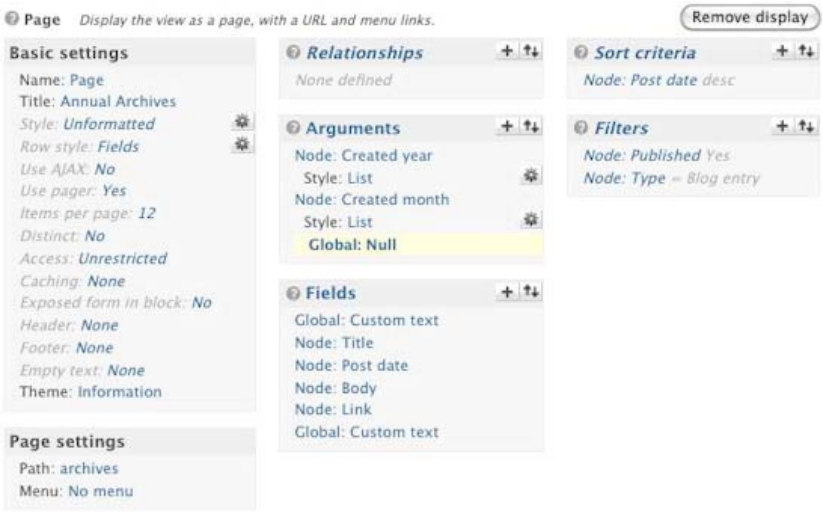


Figura 42: Exemplo de configuração de uma View em Drupal

Assim, e resumindo, só para produção de conteúdo informativo e estático no Drupal contamos pelo menos com módulos como: Editor WYSIWYG, Views, Painels, Books e Taxonomies.

"Sítio DBV": Carregamento de Informações Pessoais de Active Directory

Na secção que mostra informação referente a Docentes e Investigadores no sítio do DBV, todo o conjunto de informação presente no perfil de um professor não foi directamente inserida no Drupal por um simples editor simplificado de texto, mas sim escrita através de mecanismos de importação de informação directamente de repositórios de dados externos. No caso, a informação presente em cada perfil de um docente (Figura 43) é carregada directamente da Active Directory do Windows e do perfil de cada docente. Para esse efeito, foi utilizado um módulo Drupal denominado LDAP (http://drupal.org/project/ldap_integration) que permite o uso e sincronização de informação presente em

repositórios externos de forma relativamente simples, permitindo posteriormente a edição dessa mesma informação de forma natural através do típico editor de texto do Drupal.

Figura 43: Extracto de página com informação de Docente importado da Active Directory

Da mesma forma que para este caso foi carregada informação de uma Active Directory, este módulo também o permite fazer com outro tipo de fontes de dados externas como tipicamente bases de dados MySQL.

"Sítio DBV": Páginas que Possibilitam Download de Ficheiros

Nas páginas referentes a alunos e que têm como objectivo fornecer todo o tipo de informação a estes, é possível o descarregamento de ficheiros com informação relevante, por parte dos utilizadores do sítio Web. Para que isto seja possível de implementar via Drupal é apenas necessário o simples uso de um módulo denominado "Upload" (http://drupal.org/handbook/modules/upload) e que já se encontra instalado de origem na própria ferramenta. Através deste módulo é assim possível carregar ficheiros para páginas Drupal e disponibilizar o seu descarregamento por parte dos utilizadores.

"Sítio DBV": Sistema de Pesquisa

No canto superior do Sitio Web do DBV, logo a seguir ao cabeçalho, encontra-se um campo para pesquisa de informação em todo o sítio. No Drupal esta funcionalidade já vem instalada de origem, vindo inclusivamente e normalmente, já implementado no próprio Sítio Web. Esse mecanismo de pesquisa permite a funcionalidade típica de pesquisar conteúdos do sítio através de palavras-chave. Naturalmente que para além deste módulo de "Search" (http://drupal.org/handbook/modules/search) existem também outros que permitem obter o mesmo efeito.

"Sítio DBV": Calendário para Consulta de Eventos

Por baixo do sistema de pesquisa encontrado no Sítio Web, é possível de verificar também a presença de um calendário que permite consulta de eventos marcados para os diferentes dias do mês (Figura 44), em que um clique num dia com marcação, leva-nos a uma página com informação sobre esse evento. A implementação deste tipo de conteúdo faz-se através da instalação e uso do módulo "Calendar" (http://drupal.org/project/calendar) que permite a configuração visual de uma calendário e a agregação de conteúdos ao mesmo sobre a forma de eventos. O uso deste módulo é muito simples.

Figura 44: Imagem do Sítio DBV com sistema de pesquisa e calendário visíveis

"Sítio DBV": Sistema de Autentificação para Alunos

Outra funcionalidade bem visível no Sítio DBV, é a de um sistema de autentificação presente do lado direito das páginas (Figura 45), e que permite que alunos membros do departamento se possam autentificar no Sítio Web e aceder e alterar uma página pessoal automaticamente criada aquando do primeiro acesso. A implementação desta funcionalidade é muito simples e faz novamente uso do módulo LDAP (http://drupal.org/project/ldap_integration). Qualquer tentativa de autentificação no Sítio Web é seguida de uma verificação das credencias introduzidas e consequente comparação com credenciais registadas no repositório de alunos, professores e investigadores na Active Directory. Quando encontrado registo semelhante, é autorizada a autentificação ao utilizador.

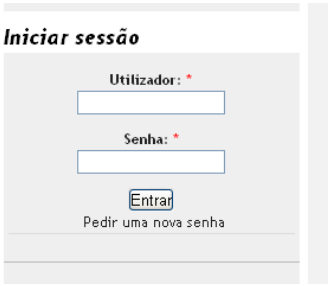


Figura 45: Imagem do Sistema de Login no Sítio DBV

"Sítio DBV": Formulário para Envio de Emails para o Departamento

Na zona de "Contactos" do sítio construído para o DBV foi implementado um pequeno formulário quer permite a qualquer utilizador do sítio, contactar a gestão do Departamento, através da escrita de mensagens que são enviadas por correio electrónico. O formulário que permite esta funcionalidade contou com o uso de dois módulos distintos, o primeiro responsável pela criação do próprio formulário com os respectivos campos, dá pelo nome de "WebForm" (http://drupal.org/project/webform) e baseia-se na criação simplificada e gráfica de formulários arrastando componentes que se pretendem criar para o interior do formulário e configurando essas componentes com as características dos campos desejados (Figura 46). A segunda baseia-se na utilização de outro pequeno módulo denominado "Captcha" (http://drupal.org/project/captcha) para verificação de identidade por parte de quem preenche o formulário, e verificação de que este se trata de facto de um utilizador humano, e não de um qualquer mecanismo automático.

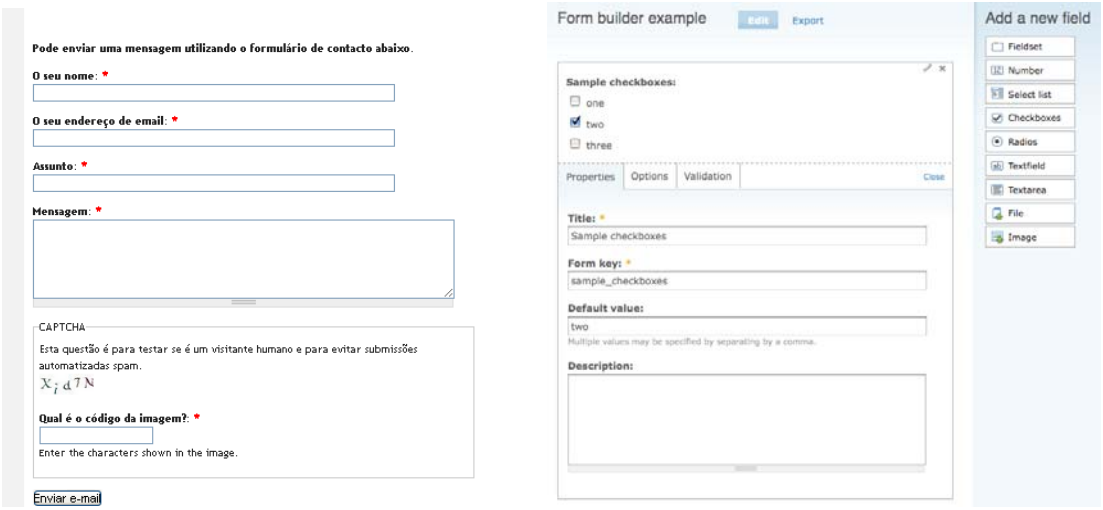


Figura 46: Formulário no Sítio DBV e módulo WebForm para Drupal

"Sítio DBV" e "Sítio CI-FCUL": Desenho e Desenvolvimento de Templates

Nos dois sítios Web que englobam este caso de estudo, houve trabalho de implementação e desenho de templates para estruturação das melhores formas de apresentação da informação. Este desenho baseia-se no Drupal em duas formas típicas: a primeira é na utilização de um de milhares de templates já existentes para uso na ferramenta e consequente alteração de pormenores através do acesso e modificação do seu código fonte, através de programação PHP e CSS, e a segundo é na implementação de raiz de um novo template, caso este muito pouca vezes utilizado, uma vez que a utilização de um template simples e sua alteração permitem poupar muito tempo em relação à criação de raiz do mesmo. Assim sendo, na criação de ambos os sítios Web, fez-se primeiro uma escolha de um template já existente, após a qual, se acedeu ao código de CSS e PHP para redefinir cores, cabeçalhos, rodapés, pormenores de estrutura, tipos de letra, etc.

Estudo de Outras Funcionalidades Implementáveis em Drupal

Para além das funcionalidades implementadas, o Drupal permite ainda o uso de muitos outros módulos responsáveis pela implementação de uma grande variedade de funcionalidades, capazes de fazerem desta ferramenta, um meio de desenvolvimento muito poderoso. Passo a descrever essas funcionalidades:

Criação de Tipos de Conteúdos Personalizados

Uma das grandes capacidades do Drupal, é o facto de através de um módulo denominado CCK (Kit de Construção de Conteúdos - http://drupal.org/project/cck) ser possível criar qualquer tipo de conteúdo pretendido por um desenvolvedor, com o número e o tipo de campos pretendido, sendo possível depois a sua vasta utilização na aplicação, criando conteúdos personalizados através do simples preenchimento dos campos que os constituem.

Módulo de Memória para Tornar Páginas Menos Pesadas

Uma vez que todas as páginas Drupal são puro PHP, poderão tender a ficar bastante pesadas, à medida que vão sendo enriquecidas de funcionalidades. Por essa razão, é essencial para a eficiência na utilização das mesmas, o uso de alguns módulos de memória que, através de pequenas caches e suas configurações, permitem guardar em memória a informação mais vezes utilizada e descartar a que é menos vezes acedida. Assim com a utilização vários módulos Drupal, é possível tornar os Sítios Web mais rápidos. Exemplos típicos desses módulos são por exemplo: Memória de Autentificação Pessoal (http://drupal.org/project/authcache), Memória Avançada (http://drupal.org/project/advcache) e Acções de Memória (http://drupal.org/project/cache_actions).

Uso de WebServices no Drupal

Nenhuma ferramenta de criação de Aplicações e Serviços Web é boa se não permitir a integração e uso de WebServices externos nas aplicações que implementa, sendo este factor essencial, para uma total integração de novos serviços com serviços já existentes na mesma instituição. O Drupal permite essa integração através do uso de um módulo denominado Serviços (Services - http://drupal.org/project/services) que através de um uso simplificado permite a invocação de URLs de outros serviços Web e recepção das suas respostas com consequente integração nos conteúdos da aplicação.

Outras Funcionalidades Possíveis em Drupal

Existe uma enorme quantidade de funcionalidades implementáveis em Drupal e que ainda não foram identificadas neste documento. No entanto, e para não tornar este estudo demasiado extenso, serão

de seguida identificadas mais algumas funcionalidades de importante referência para o desenvolvimento de aplicações no CI-FCUL, sobre a forma de um quadro básico:

Módulo Drupal a Utilizar	Observações Sobre Funcionalidade Que Implementa
Ubercart - http://drupal.org/project/ubercart	Implementação de um Carrinho de Compras no Sítio Web
PathAuto - http://drupal.org/project/pathauto	Resolução automática de caminhos URL de acordo com o título dado a cada conteúdo.
SimpleNews - http://drupal.org/project/simplenews	Implementação de um sistema de notícias por correio electrónico para membros registados no sítio Web ou em grupos pertencentes a este.
Rules - http://drupal.org/project/rules	Definição de regras ou fluxos de operações a realizar após a execução de operações específicas num sítio Web. Permite automatizar acções, e desencadear sequências de eventos.
Site Map - http://drupal.org/project/site_map	Implementação de um mapa geral do sítio Web, com navegação automática por todas as páginas.
phpCas - http://drupal.org/project/cas	Permite integrar sistemas de login único em páginas Web desenvolvidas em Drupal (PHP).
Nice Menus - http://drupal.org/project/nice_menus	Implementação de menus “deslizantes” ao nível da interface.
Content Acess - http://drupal.org/project/content_access	Definição de regras de acesso a conteúdos baseadas em autores e perfis de utilizadores.
Voting API - http://drupal.org/project/votingapi	Implementação de sistemas de votos sobre conteúdos presentes no Sítio Web desenvolvido.

Tabela 9: Quadro de outras funcionalidades implementáveis Drupal

Esta lista serve apenas para avançar mais alguns exemplos de funcionalidades possíveis de implementar, representado apenas uma reduzida parte de possibilidades, sendo um dado certo que em Drupal, é possível implementar praticamente qualquer funcionalidade que um desenvolvedor pretenda. Caso não exista um módulo, a própria comunidade encarregar-se-á de desenvolver um num curto espaço de tempo.

Estudo de Problemas e Esforço na Implementação do Caso de Estudo

Após toda a descrição da implementação dos casos de estudo, interessa especificar todo o tipo de problemas encontrados durante a realização dos mesmos, bem como de alguma forma medir o esforço verificado no seu desenvolvimento. Só com base nessas especificações, é possível ter uma ideia real das

limitações neste tipo de desenvolvimento baseado unicamente no gestor de conteúdos Drupal e posteriormente compará-lo com a solução alternativa de desenvolvimento conjunto já abordada.

Assim, passo a identificar em primeiro lugar, pormenores relacionados com o esforço de desenvolvimento, para de seguida listar os problemas encontrados na implementação:

Pormenores de Esforço de Desenvolvimento

Para implementação e análise destes casos de estudos foi necessário:

- O uso do gestor de conteúdos Drupal, tendo sido desenvolvidos os protótipos, por mim, José Coelho e por Paulo Bastos (funcionário/desenvolvedor no CI-FCUL) em período inferior a duas semanas.

- O primeiro protótipo desenvolvido em Drupal, referente ao Sítio do DBV foi desenvolvido em tempo equivalente a duas semanas, tendo sido o seu desenvolvimento levado a cabo pelo Paulo Bastos há cerca de um ano e já possuindo algum conhecimento sobre o funcionamento da ferramenta.

- O segundo protótipo desenvolvido em Drupal, foi implementado num período de dois dias apenas, sendo que a maior parte desse tempo, esteve relacionado com a minha percepção e aprendizagem inicial de pormenores da ferramenta.

- A instalação, configuração inicial e início de desenvolvimento na ferramenta Drupal, foi feita de forma muito simples e intuitiva, havendo grande quantidade de documentos de suporte para esclarecimento de possíveis dúvidas na sua instalação e configuração.

- No que respeita ao esclarecimento de dúvidas de configuração e uso da ferramenta de desenvolvimento, contei com o apoio do Paulo Bastos já conhecedor das características e pormenores da ferramenta, tendo sido bastante reduzido, desta forma, o tempo inicial de aprendizagem, já por si relativamente curto. O facto de haver uma individualidade já com conhecimentos prévios nesta ferramenta representa uma ponto favorável à sua utilização.

Problemas Verificados no Desenvolvimento:

Após um período inicial de experimentação e utilização das componentes de desenvolvimento que a ferramenta oferece, fica a ideia de que esta se apresenta bastante completa e que não há qualquer tipo de funcionalidade que não seja possível de ser desenvolvida em aplicações utilizando o Drupal. Assim sendo a identificação de problemas no desenvolvimento é felizmente difícil, uma vez que é necessário recorrer a pormenores para conseguir evidenciar falhas, a não ser os problemas relacionados com opiniões contrastantes ao uso da ferramenta como única impulsionadora de todo o desenvolvimento. Assim sendo:

- Verificou-se alguma falta de tempo para implementação de um caso de estudo de raiz em Drupal, devido ao principal foco no desenvolvimento de um protótipo na plataforma de desenvolvimento DIF2.0 e devido à não consideração da possibilidade de desenvolvimento centrada únicamente na

ferramenta de gestão de conteúdos, aquando do início do projecto. Esta não consideração deveu-se a um desconhecimento das capacidades técnicas que ferramentas de gestão de conteúdos podem deter.

- Verificou-se alguma dificuldade na configuração inicial de um sistema de sítios Drupal baseada apenas numa única instalação da ferramenta, tendo sido experimentados vários métodos, antes de por fim se concluir sobre a sequência de acções a realizar para a criação de cada Sítio Web novo que se pretende desenvolver.

- O facto de existir grande número de módulos, e de existir muitas vezes, mais que um módulo para atingir o mesmo fim, e portanto para implementar a mesma funcionalidade, cria alguma confusão aquando da pesquisa do melhor módulo, e obriga a uma experimentação dos diferentes, para conclusão de qual o melhor e mais eficiente para atingir o que se pretende. Isto resulta em alguma perda de tempo e pode levar a algumas incongruências na gestão da ferramenta sempre que são alterados módulos diferentes para funcionalidades semelhantes.

Conclusões sobre Caso de Estudo

Conclui-se da realização deste caso de estudo e de todos os pormenores da sua realização, bem como dos problemas levantados e do esforço de desenvolvimento verificado, que o desenvolvimento de aplicações com base única no gestor de conteúdo Drupal, é uma possibilidade que deve ser tomada fortemente em conta para o processo de desenvolvimento futuro a implementar no Centro de Informática da FCUL. Ao contrário do que foi verificado com o uso da plataforma de desenvolvimento no desenvolvimento conjunto de aplicações, no Drupal há já uma clara ideia de todo o tipo de conteúdos e funcionalidades que serão possíveis de implementar em aplicações e serviços que sejam seu produto, e que naturalmente um maior uso e experimentação da ferramenta apenas irá servir para criar um maior convencimento nas capacidades da ferramenta, e contribuir para que esta hipótese de desenvolvimento fique cada vez mais próximo da total adopção por parte do Centro de Informática.

Como resultado da utilização única desta ferramenta está ainda provado, que o desenvolvimento de aplicações e serviços, passará a ser uma tarefa ao alcance de uma maior número de pessoas, reduzindo a dependência para programadores, e podendo libertar estes para tarefas de implementação mais específicas e complexas. Para além disso, as tarefas de gestão simples de conteúdos podem passar a ser realizadas inclusive por individualidades que originalmente não fazem parte da equipa de desenvolvimento e que em condições presentes, simplesmente apresentam as suas ideias aos membros integrantes, para que estes depois passem à sua implementação.

A juntar a todos estes factores positivos, observa-se que a existência de uma comunidade muito extensa e activa em redor da utilização do Drupal, e a constante evolução da ferramenta de forma a acompanhar todos os desafios evolutivos da Internet, com a existência de inúmera e eficaz documentação e de uma constante distribuição e debate de conhecimentos, tornam este gestor de conteúdos numa ferramenta muito coesa, segura, compacta e acima de tudo atractiva.

Tendo em conta o ambiente que rodeia esta Instituição tende-se a concluir que quanto mais simples e acessível for o processo de desenvolvimento de conteúdos, melhor e mais eficaz será o funcionamento de todos os serviços, e que por isso, a utilização exclusiva do Drupal como ferramenta única de desenvolvimento poderá ser a solução ideal a adoptar no CI-FCUL.

.

5.3 Considerações Finais

Como conclusão final, intensifico a observação de que o desenvolvimento conjunto na DIF2.0 e no Drupal poderá servir como forma de desenvolvimento ao Centro de Informática, mas no entanto, apenas será um desenvolvimento ideal, se a plataforma de desenvolvimento for alvo de alguns melhoramentos e algumas extensões, e sobretudo se a empresa responsável pelo seu desenvolvimento, disponibilizar de forma eficaz documentos que possam suportar de melhor forma os processos de implementação de projectos. Para além disso, a aprendizagem e o aprofundamento de conhecimento nas duas ferramentas por parte de todos os integrantes da equipa de desenvolvimento, é essencial para um desenvolvimento eficaz de aplicações e serviços por parte do Centro de Informática da FCUL.

Também como conclusão final, reafirmo que o desenvolvimento no CI-FCUL baseado exclusivamente no uso da ferramenta de gestão e criação de conteúdos, Drupal, será uma solução muito boa e segura, e certamente que uma maior aprendizagem e aprofundamento de conhecimento na ferramenta por parte de todos os integrantes da equipa de desenvolvimento, bem como por individualidades de alguma forma relacionadas com os conteúdos desenvolvidos na FCUL, será possivelmente o passo que falta para um desenvolvimento mais eficaz de aplicações e serviços por parte do Centro de Informática da FCUL.

Assim sendo, e tendo duas possíveis opções para desenvolvimento futuro no CI-FCUL é descrito no capítulo seguinte, a forma como as ferramentas a utilizar podem ser mapeadas na metodologia de desenvolvimento também já definida, para que o desenvolvimento de aplicações e serviços seja um resultado tanto de boas práticas como de boas ferramentas e da correcta colaboração entre as duas.

Capítulo 6 Mapeamento das Ferramentas de Trabalho na Metodologia Adoptada

Em todo este documento foram abordadas ao pormenor várias metodologias ágeis de desenvolvimento e foi definida com base nestas, uma metodologia, e respectivo processo de desenvolvimento, adequada ao Centro de Informática da FCUL, tal como foi realizado um estudo aprofundado de ferramentas de desenvolvimento (inclusivamente com realização de casos de estudo práticos), que poderiam ser englobadas em toda essa metodologia, de forma a simplificar o desenvolvimento de aplicações e serviços Web por parte do CI-FCUL.

No entanto, e após conclusão de todos estes estudos, e após serem tiradas conclusões sobre quais os constituintes melhores para adopção na Instituição, não foi ainda especificada a forma como as diferentes ferramentas a usar no desenvolvimento, podem ser integradas na própria metodologia de desenvolvimento definida. Esse mapeamento das ferramentas de trabalho na metodologia, é essencial à real adopção da mesma e é impulsionador de um processo de desenvolvimento estruturado e disciplinado, tornando-o impossível de não seguir ou aplicar.

Neste capítulo evidenciam-se pormenores de mapeamento possíveis para cada um dos tipos de desenvolvimento abordados, e portanto define-se a forma como as ferramentas de desenvolvimento podem ser directa ou indirectamente integradas na própria metodologia de desenvolvimento.

6.1 Mapeamento das Ferramentas de Trabalho na Metodologia Adoptada – DIF2.0

A utilização da plataforma de desenvolvimento rápido de aplicações DIF2.0 é um processo complexo e à primeira vista sem qualquer relação directa com a metodologia de desenvolvimento adoptada. Diria que o único (e ao dizer único, não é meu objectivo dar-lhe pouca importância) papel da plataforma é o de auxiliar o desenvolvimento das funcionalidades inicialmente identificadas como necessárias de implementar na aplicação ou serviço, pelo Cliente, aquando do processo de recolha de requisitos.

Assim sendo, as funcionalidades presentes nos user stories que são escritos e recolhidos na fase de arranque do projecto, vão ter que se traduzir de alguma forma no interior da DIF2.0, em funcionalidades claras e por fim em serviços bem definidos. O mapeamento entre essa recolha de requisitos e o próprio uso da plataforma de desenvolvimento, para implementação das funcionalidades identificadas, é feito de forma indirecta, através do Documento Técnico de "Plano do Projecto" e do seu subdocumento "Plano Técnico" (ver capítulo 3.3 Documentos Gerados com a Metodologia Adoptada:) e ainda mais especificamente através da definição da estrutura de implementação do Projecto:

6.1.1 Relação entre "Plano Técnico" e User Stories

Uma vez que na DIF2.0 existem vários tipos de identidades que definem tipicamente a implementação de Serviços, Aplicações e Funcionalidades, estas identidades terão uma relação directa com os "User Stories" que identificam as funcionalidades a desenvolver no Projecto. Assim sendo a própria escrita deste documento é feita aquando do início do uso da plataforma, e constitui assim, o arranque da implementação do projecto, sendo definidas de forma rigorosa, grande parte das classes a implementar no projecto, e obtendo-se a estrutura geral de todo o projecto (mesmo sendo verdade que apenas são definidas as classes principais, enquanto que as classes que servem de auxílio a estas acabam apenas por ser criadas aquando da codificação das várias funções).

Assim sendo, o documento identificado como "Plano Técnico" e que faz parte do documento geral de "Plano do Projecto" mapeará os "user stories" e funcionalidades requeridas para o projecto com base na plataforma DIF2.0 da seguinte forma:

1. Estruturação de Funcionalidades: A principal forma de mapeamento presente no documento é a tabela de estruturação de funcionalidades. Nesta tabela, como já foi referido brevemente em capítulos anteriores, é identificada a relação entre cada "user story" (e portanto cada funcionalidade) e os objectos de implementação da plataforma de desenvolvimento DIF2.0. Isto é, sabendo-se que para implementação de funcionalidades ao nível da plataforma, é necessária a definição de diversos tipos de objectos - Providers, Aplicações, Serviços e Stages - esta tabela identificada para cada "user story" quais os objectos em que este está implementado. Como exemplo, verifiquemos que na figura seguinte (Figura 47) a funcionalidade marcada com o "user story id" de RC14 e que se refere ao acto de registo de uma nova conferência na aplicação a desenvolver, irá ser mapeado numa "Stage" com o nome de "FormRegConf", que por sua vez pertencerá a um serviço denominado "FormRegConfService" que pertence também à aplicação "RegistoConferenciasAplication". Existe ainda um outro campo denominado "Outros Objectos a Criar" que pretende contemplar a identificação de objectos relacionados com a implementação de uma funcionalidade, mas que não estão directamente "cravados" na estrutura da DIF2.0, ou seja, que não são de nenhum dos tipos de objecto referidos na estrutura obrigatória (no caso do registo de conferências, o objecto Conferência não é nem uma aplicação, nem um serviço, nem uma stage, mas é uma classe Java que insere e lê conteúdos de uma base de dados e que é indispensável à implementação da funcionalidade.)

Desta forma, é possível com base na escrita e consulta desta tabela (Figura 47), garantir que tanto a metodologia como as ferramentas adoptadas para o desenvolvimento futuro de aplicações e serviços, são contempladas e utilizadas de forma simultânea e relacionada, e que nenhuma das duas existe em separado da outra.

Estruturação de Funcionalidades - DIF2.0:						
Funcionalidade		Objectos de Implementação				
User Story Id	Descrição da Funcionalidade	Nome da Aplicação	Nome do Serviço	Nome do Stage	Outros Objectos a Criar	Pormenores de Implementação
USID	DESCRICAO	QQCOISAAplication	QQCOISAService	QQCOISA	NOME - TIPO	OBSERVAÇÕES
RC14	Registar uma Conferência	RegistoConferenciasAplication	FormRegConfService	FormRegConf	Conferencia - Classe	- Conferência tem de ser definida ao nível da base de dados - Classe Conferência é gerada automáticamente pela DIF2.0 com reverse engeneering
RC15	Efectuar Login no Sítio de Marcação de Conferências	RegistoConferenciasAplication	LoginRegConfService	LoginRegConf	N/A	- Página de Sucesso de Login é a Home - Página de Falha de Login é a mesma com mostragem de erro.
RC01	Apresentar Informação sobre registo de conferências	RegistoConferenciasAplication	HomeRegConfService	HomeRegConf	- Classe acessória responsável pela importação de conteúdos de outras páginas Web	
...	...	...	...	...	...	..

Figura 47: Estruturação de Funcionalidades com mapeamento entre "User stories" e Objectos de Implementação DIF2.0

2. Diagrama de Classes: Como tipicamente em qualquer diagrama de classes, é dada uma visão geral de todas as classes a desenvolver na implementação. Neste caso, incluirá a estrutura típica de qualquer aplicação desenvolvida nesta plataforma, definindo os vários tipos de objectos a criar, desde o nível de "aplicação", passando por "serviço" e chegando a "stage". Com base na consulta de um diagrama deste tipo, qualquer desenvolvedor adquire uma ideia geral da estrutura programatória de toda a aplicação a desenvolver, e entende o funcionamento geral da mesma. É de referir ainda que a elaboração de um diagrama deste género será sempre baseada na consulta e na transcrição de algum do conteúdo presente na tabela de "Estruturação de Funcionalidades". Ambas devem reflectir-se e estarem em acordo em todos os pormenores relacionados com as várias classes a desenvolver.

   Assim sendo, na imagem seguinte (Figura 48Figura 48) é mostrado um exemplo típico de uma diagrama de classes de um projecto desenvolvido na plataforma de desenvolvimento DIF2.0.

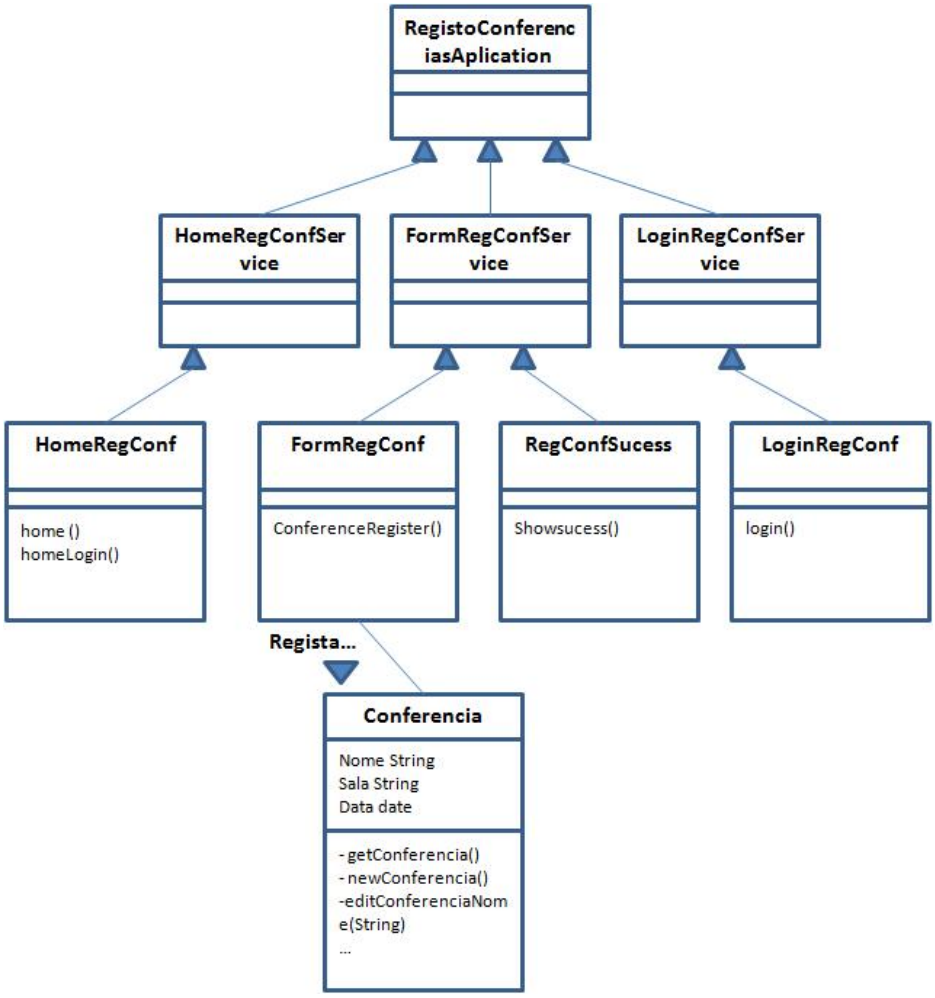


Figura 48: Excerto típico de um Diagrama de Classes que reflecte a estrutura DIF2.0

Para além destes dois tipos de registo de suporte ao desenvolvimento, também o Diagrama de Sequências faz parte deste documento técnico. No entanto este não está directamente relacionado com a ferramenta de desenvolvimento utilizada, e por isso não representa qualquer papel na relação entre a metodologia e as ferramentas adoptadas.

A única forma de relacionar esta ferramenta de trabalho com a metodologia adoptada é mesmo exclusivamente através dos seus requisitos e dos documentos acima especificados, não havendo outro ponto de contacto entre metodologia e ferramenta, terá de partir portanto sempre da própria equipa de desenvolvimento, o simultâneo uso das duas sem mais qualquer forma de estas se auto promovam, ou se sustentarem de uma forma recíproca (ao contário do que acontece no uso do Drupal, como vamos ver).

6.2 Mapeamento da Ferramenta de Trabalho na Metodologia Adoptada - Drupal

Ao contrário do que acontece no uso da plataforma de desenvolvimento DIF2.0, o Drupal permite uma relação mais directa entre o processo definido na Metodologia de Desenvolvimento e a própria implementação de funcionalidades na ferramenta de desenvolvimento. Essa relação mais directa traduz-se na utilização do Drupal como própria ferramenta de registo de toda a recolha de requisitos:

6.2.1 Relação entre "Plano Técnico" e User Stories

No caso do Drupal, a relação entre o documento de "Plano Técnico" e a forma de mapeamento dos "user stories" com as propriedades da ferramenta de gestão de conteúdos faz-se de forma bastante diferente da demonstrada com a DIF2.0. Uma vez que no Drupal não há objectos obrigatórios de programação, e não existe uma estrutura obrigatória a implementar sempre que se deseja a implementação de um serviço ou funcionalidade, a única forma de mapear os requisitos na ferramenta é através da relação directa entre os módulos utilizados para a sua implementação, e dos objectos criados para que a funcionalidade seja uma realidade. Como é visível no exemplo de tabela que se segue (Figura 49), para um "user story" identificado como RC16 (e portanto pertence ao projecto de "**R**egisto de **C**onferências") referente à visualização de notícias na aplicação, são identificados como módulos para implementação da funcionalidade, o módulo "View" e o módulo "Panels", sendo criados os objectos referentes a cada um. Desta forma o desenvolvedor tem uma ideia do tipo de objectos que terá de criar para implementar cada funcionalidade, e é feita então uma relação entre os vários requisitos da aplicação a desenvolver e os vários constituintes da ferramenta a utilizar nessa implementação.

Estruturação de Funcionalidades - Drupal:						
Funcionalidade		Módulos a Utilizar			Objectos de Implementação	
User Story Id	Descrição da Funcionalidade	Nome do Módulo	Função do Módulo	Pormenores de Implementação	Nome do Objecto	Tipo do Objecto
USID	DESCRICAO	NOME DO MODULO	FUNÇÃO DO MÓDULO	OBSERVAÇÕES	NOME	TIPO
RC16	Visualizar notícias de conferências marcadadas.	View, Panels	Views: Visualizar certo tipo de conteúdos filtrando pelas suas características. Panels: Colocar um conjunto de informações de tipos diferentes no mesmo bloco de conteúdo.	N/A	Vista_Noticias Painel_Noticias	View Panel
...	...	...	...	...	...	...

Figura 49: Estruturação de Funcionalidades com mapeamento entre "User stories" e Módulos de Implementação Drupal

Após verificação da "Estruturação de funcionalidades" nas duas ferramentas possíveis de utilizar para desenvolvimento, interessa referir que o uso simultâneo destes dois tipos de tabelas é também naturalmente uma realidade. Sempre que forem utilizadas em simultâneo as duas ferramentas serão mapeados os "user stories" nas duas também em simultâneo, sendo que nesse caso, funcionalidades mais

simples serão sempre levadas a cabo pelo Drupal enquanto mais complexas terão sempre relação com a DIF2.0.

Ao contário do que acontece na DIF2.0, no documento de "Plano Técnico" não existirão diagramas de classes que reflictam a estrutura a criar no Drupal, uma vez que seria inútil para qualquer programador a consulta de um documento desse género, e uma vez que a construção desse tipo de documento não seria uma tarefa fácil de levar a cabo á priori, e antes de qualquer pedaço de desenvolvimento ser levado a cabo na implementação. Ou seja, a utilização do Drupal não define estruturas pré-definidas como a utilização da DIF2.0 e por isso seria inútil forçar uma estrutura do género.

Para além destes dois tipos de registo de suporte ao desenvolvimento, também o Diagrama de Sequências faz parte deste documento técnico. No entanto este não está directamente relacionado com a ferramenta de desenvolvimento utilizada, e por isso não representa qualquer papel na relação entre a metodologia e as ferramentas adoptadas.

6.2.2 Módulo para Recolha de Requisitos no Drupal

Sabendo-se que o processo de recolha de requisitos se baseia na escrita de “User Stories” por parte dos Clientes de aplicação, e sabendo-se que o Drupal permite através dos seus diferentes módulos de funcionalidades, a definição de tipos de conteúdos personalizados, conclui-se que como forma ideal de recolha de requisitos, o desenvolvimento de um tipo de conteúdo denominado “User Story” permite que a própria ferramenta possa funcionar como repositório para o registo destes no processo de recolha de requisitos. Assim sendo, o membro da equipa de desenvolvimento encarregado de guiar o Cliente no processo de escrita de “User Stories”, será responsável pela escrita dos mesmos no Drupal, que no final desta fase terá uma quantidade de conteúdos criados, igual à quantidade de “User Stories” escritos pelo Cliente. Para que isto seja possível, é naturalmente necessário que um tipo de conteúdo “User Story” no Drupal apresente os campos iguais ao que este tipo de requisito define, desde o seu nome, à sua descrição, à sua prioridade e ao esforço estimado, todos estes serão campos deste tipo de conteúdo.

Para além da criação de um tipo de conteúdo deste género (tipicamente através do modulo Drupal CCK, já descrito nos Casos de Estudo de desenvolvimento em Drupal), é ainda possível o desenvolvimento de um módulo próprio da metodologia a adoptar, que permita fazer uma ordenação e apresentação destes requisitos a todos os desenvolvedores, e permita a actualização e revisão do estado de completude dos mesmos à medida que o desenvolvimento vai avançando.

Assim, a gestão e prioritização dos requisitos identificados na fase inicial do processo que define a metodologia adoptada, são feitas directamente através da própria ferramenta de desenvolvimento Drupal, estabelecendo-se uma ligação directa entre esta e a própria metodologia, que conduz a uma junção natural entre processos e implementação.

Esta criação de um módulo de recolha de requisitos é algo facilmente atingível no Drupal devido a basear-se numa implementação simples que oferece posteriormente grande facilidade de trabalho por parte de qualquer um, mas de difícil implementação na DIF2.0, uma vez que a consulta e gestão desses requisitos estaria apenas ao alcance dos programadores com conhecimentos suficientes para trabalhar na plataforma de desenvolvimento, e uma vez que à partida seria necessário o desenvolvimento de um módulo de funcionalidades capaz inserir, gerir e apagar "user stories", implementação essa que não seria uma tarefa de simples elaboração como no Drupal.

6.3 Discussão sobre Mapeamento das Ferramentas de Trabalho na Metodologia Adoptada

Ao considerar-se, acima, as duas formas de relacionar as ferramentas possíveis de utilização no desenvolvimento, com a metodologia a adoptar, percebe-se também que, tanto a perspectiva de desenvolvimento conjunto, como a perspectiva de desenvolvimento utilizando unicamente o Drupal, têm o gestor de conteúdos como ponto comum, e por isso será sempre possível o mapeamento dos requisitos do projecto como conteúdos no Drupal. No entanto, é aí que interessa considerar a forma como esse mapeamento contribui para o desenvolvimento nas duas perspectivas:

- No desenvolvimento conjunto, é marcante a divisão de tipo de tarefas por tipo de desenvolvedores, e é definido que os desenvolvedores com maiores conhecimentos programam funcionalidades servindo-se exclusivamente da plataforma de desenvolvimento DIF2.0, não chegando a fazê-lo no Drupal, que serve apenas de ferramenta de criação de conteúdos simples por parte de utilizadores com menos conhecimentos técnicos. Por esta razão, a utilização dos módulos criados no Drupal para gestão e prioritização dos recursos do projecto, não trariam um proveito assinalável para os que realizam desenvolvimento na DIF2.0, uma vez que estes teriam de passar a utilizar as duas ferramentas em simultâneo, ou teria de ser definido, em simultâneo com o levantamento de recursos no Drupal, o conteúdo do documento técnico do projecto, dificultando a forma de consulta das tarefas a realizar, e complicando a própria metodologia na fase inicial do seu processo.

- No desenvolvimento unicamente baseado em Drupal, a vantagem do mapeamento dos "User Stories" na ferramenta é enorme, uma vez que integra a própria ferramenta directamente na metodologia, fazendo com que uma seja a continuação da outra. Os desenvolvedores têm acesso constante às tarefas a realizar em todo o projecto, para além de saberem quem está encarregue do desenvolvimento das várias funcionalidades e de terem uma visão geral e constante do estado de realização de todo o projecto.

Concluindo sobre a integração das ferramentas de trabalho na metodologia de desenvolvimento adoptada, pode-se dizer que é evidente a facilidade de integração do Drupal na metodologia, e a forma como um desenvolvimento baseado exclusivamente numa das ferramentas permite simplificar a gestão do

desenvolvimento, e estabelecer uma ligação evidente com os processos da metodologia de trabalho, em contraste com o facto de um desenvolvimento constituído pela utilização de várias ferramentas em simultâneo poderá ter o efeito contrário e levar a uma completa fragmentação entre o uso das mesmas e o seguimento de um processo de desenvolvimento bem definido.

Capítulo 7 Conclusão

Após todos os processos de análise e estudo estarem finalmente concluídos neste Projecto de Engenharia Informática, interessa deixar claros alguns pontos numa pequena discussão final de todo o trabalho realizado, e das melhores opções a tomar para o futuro do desenvolvimento de aplicações e serviços no Centro de Informática da FCUL.

7.1 Sumário do trabalho realizado

Antes de mais, convém fazer uma breve referência a todo o trabalho realizado ao longo destes nove meses de projecto. Olhando para trás e para todo o conteúdo deste documento, é possível escrutinar a existência de duas diferentes temáticas no que diz respeito a Estudo de Metodologias de Desenvolvimento (capítulos 2 e 3) e no que diz respeito a Estudo de Ferramentas de Desenvolvimento (capítulos 3,4 e 5). Estas temáticas relacionam-se de forma indirecta, uma vez que ambas dizem respeito a formas de desenvolvimento a adoptar no Centro de Informática da FCUL. No entanto, também se relacionam de forma directa, sendo esse processo de ligação descrito no capítulo 6 deste documento.

Estas duas diferentes temáticas constituíram também duas diferentes épocas de trabalho no que a este projecto diz respeito, tendo sido, como é facilmente observável no capítulo de planeamento deste projecto, a primeira dedicada à analise de diversas metodologias e ao estudo de qual a que se adoptava melhor ao grupo de desenvolvimento, e a segunda a análise de ferramentas de desenvolvimento e o estudo de qual a melhor forma de desenvolvimento com base nas ferramentas adoptadas.

Assim sendo, grande parte deste projecto foi constituída por investigação de novas formas de desenvolvimento mais eficaz para o CI-FCUL, enquanto uma pequena parte (e final) foi composta pela realização prática de vários exemplos de aplicações, como forma de provar a aplicação real de tudo aquilo que foi primeiramente investigado.

Concluindo sobre todo o trabalho levado a cabo, digo ainda que este foi marcado, desde metade da sua realização, por alguma indefinição de objectivos claros do projecto, devido à constante alteração do propósito final do mesmo por parte das entidades responsáveis pela sua realização, tendo sofrido um desvio em relação ao que era inicialmente previsto, e tendo resultado numa finalização demasiado apressada para um fim, que poderia ter sido desde o início, melhor definido por parte dos responsáveis e melhor transmitido à minha pessoa. Se por parte do Centro de Informática tivesse existido uma melhor definição de todo o projecto e seus objectivos, e uma menor inconstância durante todo o projecto no que diz respeito às varias tarefas a realizar para atingir o fim pretendido, então todo este estudo estaria melhor fundamento e orientando de uma forma mais esclarecida e mais conclusiva.

7.2 Conclusão sobre Desenvolvimento Futuro no CI-FCUL

A principal conclusão a que este projecto tenta chegar é a de resolução da ausência de uma metodologia de trabalho e desenvolvimento no CI-FCUL, com a adopção de processos metodológicos de trabalho, tal como de ferramentas capazes de aumentar a eficiência e a qualidade dos produtos desenvolvidos por este Organismo da FCUL. Assim, interessa agora concluir sobre três aspectos principais: A metodologia ágil de desenvolvimento a adoptar, as ferramentas e tipo de desenvolvimento baseado nestas que mais favorece o CI-FCUL e sobre os problemas que terão de ser ultrapassados para que a adopção destas novas forma de trabalho e desenvolvimento sejam de facto uma realidade.

7.2.1 Conclusão sobre Metodologia de Desenvolvimento a Adoptar

Do estudo pormenorizado de metodologias de desenvolvimento no capítulo 2 deste documento, e consequente definição de uma metodologia de desenvolvimento no capítulo 3, conclui-se que a metodologia a adoptar pelo Centro de Informática da FCUL é uma metodologia de processos bastante simples e inteiramente baseada na simplicidade e orientada à implementação de funcionalidades identificadas pelos Clientes das próprias aplicações a desenvolverem. Com uma forte base numa das metodologias de desenvolvimento mais usadas em todo o mundo informático – O Método de Programação Extrema – e com algumas vertentes adoptadas de uma metodologia profundamente orientada ao desenvolvimento de funcionalidades – FDD -, é composta por um processo que se inicia numa recolha de requisitos inteiramente orientada ao cliente. Fazendo uso de apenas um simples documento, em que estes contam histórias de utilização, guia então de forma simplificada o desenvolvimento até uma fase de implementação orientada às diferentes funcionalidades sugeridas por essas mesmas histórias, e que de forma gradual vão sendo distribuídas pelos vários programadores ou desenvolvedores e vão sendo integradas num protótipo aplicacional. Não valendo a pena nova descrição intensa da metodologia definida no capítulo 3, valerá a pena resumir o facto de esta atribuir papéis e responsabilidades básicos aos diferentes membros do CI-FCUL e dedicar tempo à escrita de documentos durante a implementação de qualquer projecto. Documentos esses, que de alguma forma, acabam por ser os responsáveis pela organização de todos os projectos, e pela recolha de pormenores de desenvolvimento que ajudarão este núcleo de desenvolvimento a evoluir como equipa e a desenvolver produtos com cada vez maior eficiência.

A adopção da metodologia definida é no entanto, e naturalmente, da responsabilidade das personalidades que governam o CI-FCUL, tendo sido aqui apresentada, especificada e justificada uma solução que me parece capaz de fazer frente aos desafios de desenvolvimento que este grupo tem pela frente, melhorando de sobremaneira a eficiência do processo de desenvolvimento, e procurando suportar todos os projectos futuros e com eles saber sempre evoluir.

7.2.2 Conclusão sobre Ferramentas de Trabalho a Adoptar

Dos vários estudos e análises efectuados a um grande número de ferramentas de implementação de aplicações e serviços Web, concluíram-se um conjunto de pontos importantes para o futuro do CI-FCUL e que com certeza serão alvo de debate mais intenso no futuro, aquando da oportunidade de adopção das soluções apresentadas:

- O facto de não ter sido abordado um estudo prévio para determinação da plataforma de desenvolvimento ágil a adoptar no desenvolvimento, antes de se partir para a utilização da plataforma DIF2.0, representou uma barreira a uma solução final óptima, e constituíu um ponto negativo para o projecto, uma vez que após uma breve análise de plataformas alternativas, ficou claramente visível a existência de opções mais simples e eficazes para a implementação do mesmo tipo de serviços ou aplicações.

- O estudo e análise de vários gestores de conteúdos, realizado no âmbito deste projecto permite concluír que a melhor opção no que a este tipo de ferramentas diz respeito, é o Drupal, e que o desenvolvimento de conteúdos baseado na sua utilização permite inclusivamente a implementação de todo o tipo de funcionalidades que um serviço ou aplicação Web podem requerer.

- O desenvolvimento conjunto de aplicações e serviços, com base numa plataforma de desenvolvimento – DIF2.0 – e num gestor de conteúdos – Drupal – é uma possibilidade real, e uma alternativa que merece ser seriamente ponderada, dependendo no entanto de futuras extensões à plataforma em causa, ou simplesmente da verificação de um desenvolvimento eficaz após uma maior experimentação da mesma. No entanto, e da forma como presentemente a plataforma ágil de desenvolvimento se apresenta, será uma solução arriscada para desenvolvimento, uma vez que exigirá bastante tempo de aprendizagem por parte dos desenvolvedores, e uma vez que limita a apenas programadores experientes a implementação de conteúdos Web mais complexos, deixando apenas o desenvolvimento de conteúdo simples ao alcance de mão-de-obra menos especializada em técnicas de programação. Para além disto, a necessidade de integração das duas (integração essa provada como possível) aumenta a complexidade de gestão de todo o desenvolvimento, quando o que se devia pretender com a adopção de novas ferramentas era exactamente a simplificação de processos.

- O desenvolvimento de aplicações e serviços Web, baseado unicamente em Drupal é também uma possibilidade real de adopção no CI-FCUL, e embora tenha sido uma alternativa sugerida a mais de meio da execução deste projecto, figura-se presentemente como a melhor opção de desenvolvimento estudada, permitindo uma maior eficiência na implementação de aplicações Web, graças a uma implementação de funcionalidades em mais alto nível, e com menos recorrência a técnicas de programação. Isto permite que qualquer tipo de funcionalidade seja desenvolvida por qualquer tipo de utilizador da

ferramenta, possibilitando uma extensão natural da equipa de desenvolvimento a individualidades com menos conhecimentos de programação, e simplificando toda a gestão de conteúdos a apenas uma única ferramenta. Por fim e como factor que pode ser determinante, a possibilidade de introduzir funcionalidades da ferramenta no próprio processo que caracteriza a metodologia de desenvolvimento a adoptar, garante uma maior convergência no desenvolvimento, e possibilita uma maior naturalidade na adopção simultânea de ferramenta e metodologia.

7.3 Possibilidades de Trabalho Futuro no Projecto

Após a finalização deste projecto, é importante avançar com possibilidades de trabalho futuro para que a adopção das ideias formuladas seja feita de uma forma mais sustentada. Assim sendo, defino de seguida uma série de pontos a considerar numa futura extensão ou conclusão deste projecto:

- Realização da implementação de um caso de estudo de raiz na ferramenta Drupal, que permita simular um inteiro serviço fornecido pelo CI-FCUL a alunos da Faculdade, interessando a implementação total de um sítio Web e não de apenas um protótipo com pontos de desenvolvimento por concluir.

- Realização da implementação do mesmo caso de estudo utilizando conjuntamente o gestor de conteúdos e plataforma de desenvolvimento DIF2.0, para que posteriormente seja possível a execução de um estudo de esforço que permita comparar tempos de desenvolvimento das diferentes funcionalidades em cada tipo de desenvolvimento, bem como a qualidade do produto final, a capacidade de gestão dos conteúdos e a forma como a metodologia é utilizada no desenvolvimento de ambos os casos de estudo. Este estudo permitiria uma melhor comparação entre os dois tipos de desenvolvimento, podendo permitir chegar a uma conclusão final ainda mais evidente de qual a melhor alternativa para futuro.

- Realização de experiências de aplicação da metodologia de desenvolvimento adoptada a vários casos de estudo, desde o processo de levantamento de requisitos, à forma como é delineado o desenvolvimento das diferentes funcionalidades, à forma como é gerido o produto final. Estas experiências permitiriam uma melhor percepção de todas fases da metodologia, bem como permitiria um aprimoramento das várias técnicas de recolha de requisitos e de implementação de funcionalidades. Este ponto estava planeado originalmente no projecto, no entanto e devido ao surgimento da necessidade um estudo mais aprofundado de ferramentas de desenvolvimento possíveis de adoptar, tornou-se impossível de realizar em tempo útil.

- No âmbito de utilização única do gestor de conteúdos Drupal como ferramenta de desenvolvimento de aplicações e serviços, haverá a considerar no futuro o uso de outra metodologia de desenvolvimento que não a definida, ou pelo menos o uso de práticas dessa metodologia. De seu nome, método de Processo Racional Unificado (RUP), esta metodologia foi

já descrita no capítulo e nos anexos referentes a metodologias ágeis de desenvolvimento, e conta, entre outras práticas, com a definição e recolha de requisitos com base em diagramas UML e com o mapeamento desses diagramas em classes que implementam as funcionalidades de uma aplicação ou serviço, sendo naturalmente orientada ao uso de ferramentas semi-automáticas de implementação de componentes Web (caso do Drupal).

7.4 Adopção Presente e Futura de Ideias do Projecto

Para finalizar todo este projecto, e como forma de sustentar uma das opções avançadas como forma de desenvolvimento futuro de aplicações e serviços no CI-FCUL, interessa referir que nos últimos dois meses de execução deste projecto, deu-se já um início de desenvolvimento em Drupal de Aplicações Web simples, como é o caso do desenvolvimento de um sítio Web para uma fundação ligada à FCUL, e que este facto aliado ao facto de no passado terem sido já desenvolvidas outras aplicações Web do mesmo tipo para diferentes Departamentos da Instituição (como é o caso do desenvolvimento do Sítio Web abordado no caso de estudo de desenvolvimento em Drupal, levado a cabo pelo funcionário do CI-FCUL, Paulo Bastos em www.dbv.fc.ul.pt), também usando o gestor de conteúdos Drupal como ferramenta única de desenvolvimento, ajuda a sustentar a opção de desenvolvimento baseada unicamente em Drupal.

Quanto ao desenvolvimento na plataforma de desenvolvimento DIF2.0, está neste momento a iniciar-se uma experimentação mais profunda da plataforma, como o desenvolvimento de pequenos serviços na mesma, e com o início de desenvolvimento de verdadeiros serviços a adivinhar-se para os próximos meses.

É objectivo de todo este projecto, que já nos próximos projectos a iniciar por parte do CI-FCUL seja também seguida a metodologia de desenvolvimento delineada, com o uso dos modelos de documentos definidos como parte do processo de desenvolvimento, a ser o primeiro passo para a concretização deste objectivo.

Bibliografia

[Ambler, 2002] - Ambler, S. (2002). Lessons in Agility from Internet-Based Development. IEEE Software, 19(2): 66-73

[Ambler, 2002b] - Ambler, S. (2002). Agile Modeling: Effective Practices for Extreme Programming and the Unified Process, New York, John Wiley & Sons, Inc.

[Beck, 1999] - Beck, K. (1999). Extreme programming explained: Embrace change. Reading, Mass., Addison-Wesley

[Beck et al, 2001] - Beck, F., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. e Thomas, D. (2001), Manifesto for Agile Software Development (22.3.2002) http://AgileManifesto.org

[Cake, 2010] - Cake Software Foundation (2010). Sítio Web: "CakePHP: the rapid development php framework." http://cakephp.org/

[Cockburn, 1998] - Cockburn, A. (1998). Surviving Object-Oriented Projects: A Manager's Guide. Addison Wesley Longman

[Cockburn, 2002] - Cockburn, A. (2002). Agile Software Development. Boston, Addison-Wesley

[Cockburn e Williams, 2001] - Cockburn, A., Williams, L. (2001). The Costs and Benefits of Pair Programming. The XP Series. Addison-Wesley

[Digitalis, 2008] - Digitalis (2008). Sítio Web: "DIF - Digitalis Framework" http://netpa.digitalis.pt/apache2-default/dif2

[Glass, 2001] - Glass, R. L. (2001). Agile Versus Tradicional: Make Love, Not War! Cutter IT Journal 14(12): 12-18

[Haungs, 2001] - Haungs, J. (2001). Pair Programming on the C3 project. Computer 34(2): 118-119

[Hawrysh, 2000] - Hawrysh, S. e Ruprecht, J. (2000). Light Methodologies: It's Like Dejá Vu All Over Again. Cutter IT Journal. 13:4-12

[Highsmith, 2000] - Highsmith, J. A. (2000). Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing.

[Highsmith & Cockburn, 2001] - Highsmith, J., e Cockburn, A. (2001). Agile Software Development: The Business of Innovation. Computer 34(9): 120-122

[Hunt, 2000] - Hunt, A e Thomas, D. (2000). The Pragmatic Programmer. Addison Wesley

[James, 2010] - Describing the OutSystems Agile Platform Service Studio experience (19/05/2010). TechRepublic: Programming and Development. http://blogs.techrepublic.com.com/programming-and-development/?p=2564

[Khramtchenko, 2004] - Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments 5-10

[Kruchten, 2000] - Kruchten, P. (2000). The Rational Unified Process: and Introduction. Addison-Wesley

[Kumar, 2003] - Kumar, A. (2003). Extreme Programming Client Interaction And Requirements Gathering Method for Software Projects

[Kumar, 2009] - Kumar, A. (2009). Build your Project using Extreme Programming. #2 of a Series

[McCauley, 2001] - McCauley, R (2001) Agile Development Methods Poised to Upset Status Quo. SIGCSE Bulletin 33(4): 14-15

[Miller, 2001] - Miller, D e Lee, J. (2001). The people make the process: commitment to employees, decision making, and performance. Journal of Management 27: 163-189

[OutSystems, 2008] - OutSystems Agile Platform Solution Overview (2008).

[Palmer e Felsing, 2002] - Palmer, S. R. e Felsing, J. M. (2002). A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall

[Schwaber, 1995] - Schwaber, K. (1995). Scrum Development Process. OOPSLA'95 Workshop on Business Object Design and Implementation. Spinger-Verlag

[Schwaber e Beedle, 2002] - Schwaber, K. e Beedle, M. (2002). Agile Software Development With Scrum. Upper Saddle River, NJ, Prentice-Hall

[Serena, 2007] - Serena Software Inc. (2007) An Introduction to Agile Software Development - by Serena

[Stapleton, 1997] - Stapleton, J. (1997). Dynamic systems development method - The method in practice. Addison Wesley

[Stojanovic et al., 2003] - Stojanovic, N., Hartmann, J. e Gonzalez, J.(2003). Ontomanager - a system for usage-based ontology management. In Proc. of FGML Workshop. SIG of German Information Society (FGML).

[Terra, 2002] - Terra, J. (2002). Portais Corporativos - A Revolução na Gestão do Conhecimento, São Paulo, Negócio Editora

[Williams, 2007] - Williams, L (2007) A Survey of Agile Development Methodologies

Apêndice A

Análise de Metodologias Ágeis de Desenvolvimento: Estudo para o Futuro do CI

Antes de referir qualquer metodologia, convém ter em mente que uma única metodologia não será capaz de cobrir todo o espectro de diferentes tipos de projecto, ou todo o tipo de propósitos ***[Hawrysh, 2000; Glass. 2001]***. A orientação de uma certa instituição para um certo tipo de projectos poderá ajudar a ditar toda a escolha do método que se procura ser o mais eficaz para além do mais abrangente.

É com esta lógica em mente que de seguida passo a abordar ao pormenor vários métodos fiéis aos princípios de processos ágeis de desenvolvimento de software, abordando para cada um:

- Os **processos** (e práticas) que o definem: descrição das fases que constituem o ciclo de vida de produção de software;
- Os **papéis** – responsabilidades - que a metodologia exige da equipa de desenvolvimento;
- As formas de **adopção, experiencias e práticas** do método: relatos de experiências, identificação de práticas comuns (actividades concretas que um método define para uso no processo de desenvolvimento) e limitações de uso do método;
- Probabilidades de **aceitação no CI**: principais características que refutam ou elegem o método para uso no CI.

Programação Extrema (Extreme Programming) – XP

Segundo Haungs ***[Haungs, 2001]*** a Programação Extrema é "uma oportunidade de garantir que o trabalho é feito", consistindo num agrupamento de práticas consideradas eficazes em todo o tipo de desenvolvimentos de software levados a cabo na última década ***[Stojanovic et al, 2003].*** Constituindo-se como uma metodologia que surgiu após se constatar a existência de grande número de problemas com ciclos de desenvolvimento muito longos em modelos tradicionais de desenvolvimento (o modelo "cascata" já falado).

Sendo, como referi, um conjunto de "boas práticas" de desenvolvimento, faz todo o sentido enumerar e explicar alguns desses costumes, que contribuem em maior escala para a definição do que é o método de Programação Extrema:

- **"Jogo de Planeamento":** Permite, através da combinação de prioridades de negócio - definidas pelo cliente, por factores externos como o ambiente, ou pelos próprios elementos da equipa de desenvolvimento -, e de estimativas técnicas das funcionalidades a implementar no sistema, determinar a dimensão da próxima versão do produto a ser lançada.
- **Pequenas Incrementações**: Fazer com que um produto entre em produção desde o início do seu desenvolvimento, e possibilitar o lançamento contínuo de novas versões em curtos intervalos de tempo.
- **Metáfora Guia**: Guiar todo o desenvolvimento de um produto, através da partilha de uma história simples descritiva do funcionamento total do sistema.

- **Simplicidade no Desenho**: Produto deve ser desenhado da forma mais simples possível, removendo complexidade à medida que esta vai sendo descoberta.
- **"Testing"**: Escrita contínua de testes de unidade. Aprovação nos testes possibilita continuidade das iterações, chumbo implica redesenho da funcionalidade. Cabe aos clientes efectuar a demonstração de cada funcionalidade implementada.
- **"Refactoring"**: Qualquer alteração implementada no sistema em desenvolvimento é sempre feita sem alteração do seu (bom) funcionamento e tendo em vista a remoção de duplicidade, o melhoramento comunicacional, a simplificação do projecto, ou a adição de flexibilidade.
- **"Pair Programming" (programação em pares):** Trabalho desenvolvido em espaço aberto com posicionamento da equipa a toda a volta, fazendo com que o núcleo de desenvolvimento de cada projecto, seja constituído por um par de programadores que trabalham colaborativamente no mesmo algoritmo, ou tarefa de programação, sentando-se lado a lado no mesmo computador. ***[Cockburn e Williams, 2001]***
- **Propriedade Colectiva de Código:** Qualquer pessoa pode alterar qualquer pedaço de código a qualquer altura do projecto.
- **Integração Contínua:** Integração de dados no sistema é feita várias vezes ao dia, sempre que uma tarefa é finalizada ou que uma funcionalidade é concluída.
- **Semana de 40 horas de trabalho:** Semanas de trabalho nunca passam este limite em duas semanas consecutivas.
- **Presença do cliente no Local:** Incluir um utilizador real na equipa, sempre disponível para responder a qualquer questão, para que os objectivos da aplicação sejam cumpridos.
- **Standards de Código:** Código é escrito por todos os programadores, seguindo regras bem definidas e dando sempre ênfase à comunicação durante todo o processo de codificação.

Todas as práticas agora definidas fazem parte do método de Extrema Programação funcionando e combinando-se de uma forma inovadora e sobretudo funcional. Sobre quase todas as práticas, recai um peso grande de regras de programação, fazendo com que este método atribua aos programadores o poder necessário para responderem com confiança, e de forma eficaz, a requisitos que se encontram permanentemente em mudança fruto de grande indecisão, insegurança e ingenuidade dos clientes.

Processos XP:

A identificação do ciclo de vida de todo o processo de Programação Extrema impõe-se então para uma percepção mais concreta do funcionamento da metodologia. São identificáveis 6 fases bem distintas em todo o processo de desenvolvimento (Figura 50): ***[Beck, 1999]*.**

Exploração: Como o próprio nome indica, esta fase marca o arranque do processo com a identificação exploratória das componentes a desenvolver e dos objectivos a serem correspondidos pelo

sistema a desenvolver. Esse levantamento inicial de requisitos é neste caso levado a cabo pelos próprios clientes do projecto, que escrevem as funcionalidades que pretendem incluir na primeira versão da aplicação. A exploração engloba também toda a equipa do projecto e sua familiarização com as ferramentas a serem utilizadas bem como com os requisitos identificados pelos clientes. Nesta etapa de lançamento de desenvolvimento a duração é de poucas semanas a poucos meses.

Planeamento: Após definição inicial, é de seguida necessário alinhavar requisitos de forma planeada, definindo a ordem de prioridades para as funcionalidades escritas pelos clientes e estimando o tempo de implementação das mesmas. Programadores estimam assim o esforço necessário para implementar e desenhar cada funcionalidade e chegam a um acordo sobre os conteúdos da primeira versão. Toda esta etapa é relativamente curta traduzindo-se numa duração de poucos dias.

Período de Iterações (até Lançamento): Como fase natural a ocorrer após exploração e planeamento, segue-se um período bem estipulado de iterações com o objectivo de implementar cada uma das funcionalidades definidas anteriormente que culminarão no lançamento de uma versão do sistema. Este plano de iterações divide-se num número de limitado de iterações, em que cada uma levará de 1 a 4 semanas a desenvolver. A primeira iteração cria o sistema (na sua generalidade) e sua arquitectura, seleccionando as funcionalidades sobre a qual este se apoia. Cabe sempre ao cliente a selecção das funcionalidades para cada iteração. Toda esta fase de implementação é inteiramente baseada num método denominado “Pair Programming” (Programação em Pares).

Produção: Uma vez implementadas as funcionalidades, é importante estas serem testadas e verificadas em termos de performance no sistema. Nesta fase são levadas a cabo todas estas tarefas, garantindo-se que no final o sistema ficará pronto para produção. Ideias e sugestões emergentes nesta fase são documentadas para implementação futura em fase de manutenção.

Manutenção: Fase final da metodologia que no fundo repete todas as fases descritas até agora. Sistema é mantido em produção enquanto se produzem novas iterações para responderem a novas funcionalidades escritas pelos clientes, levando-se a novas versões do produto. Para que seja efectuado com sucesso esta fase pode necessitar de incorporar número adicional de desenvolvedores e/ou mudar a estrutura da equipa.

Morte: Considerada uma fase, embora não o seja realmente. É atingida quando não existem mais funcionalidades alvo de implementação por parte do cliente. Nesta fase em que o sistema está já concluído, toda a documentação relacionada com este é escrita. A Morte do sistema também pode ocorrer em situações em que o sucesso esperado não é atingido ou quando todo o custo do projecto se torna demasiado caro para que a produção seja continuada.

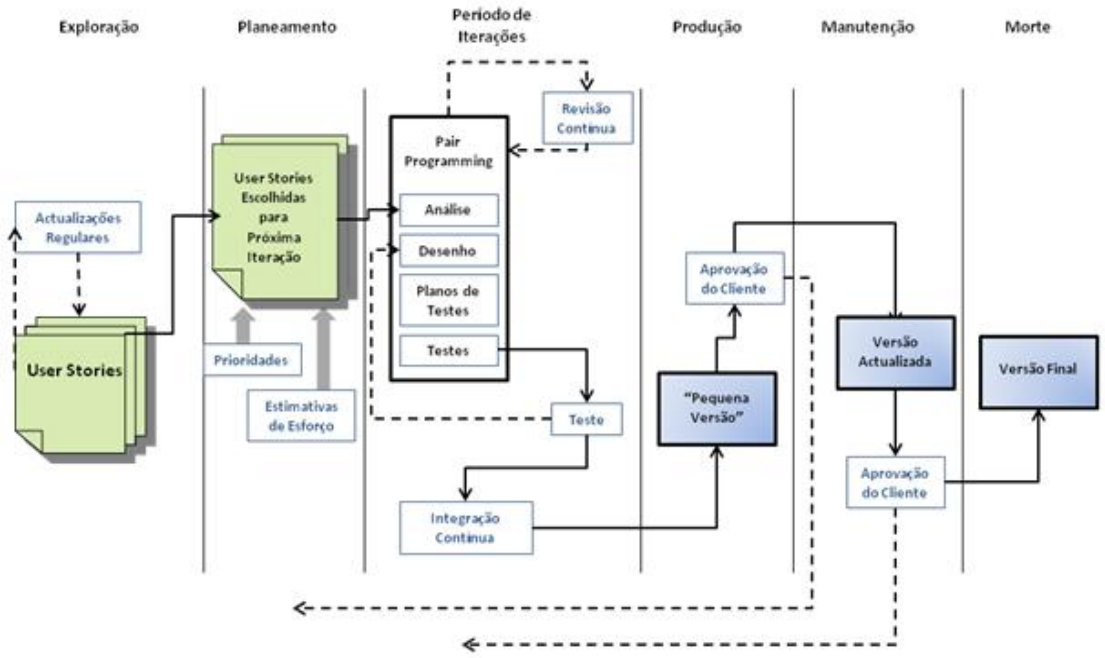


Figura 50: Esquema do Processo de Desenvolvimento do XP

Papéis e Responsabilidades XP:

Para que uma metodologia de desenvolvimento como o XP seja bem delineada, e para que seja possível atingir o sucesso é necessário dividir responsabilidades e definir papéis específicos dentro da equipa de desenvolvimento. Só desta forma é possível garantir que todos os elementos conhecem as tarefas que terão de executar sem entrar em conflito com outros elementos da mesma equipa e que gradualmente o sistema verá adicionadas funcionalidades pensadas numa perspectiva geral, e implementadas de forma independente ou por responsáveis distintos.

Definem-se assim os seguintes papéis sobre os elementos da equipa de desenvolvimento:

Programador: Membro da equipa responsável por escrever testes bem como o código da aplicação em desenvolvimento. É da sua responsabilidade manter o código do programa tão simples e definido quanto possível. Para que o projecto seja possível é essencial existir comunicação e coordenação entre os vários programadores da equipa.

Cliente: Indivíduo responsável (na maior parte das vezes) pelo nascimento do projecto, e aquele que em primeiro lugar tem a ideia sobre o sistema a construir e que na maior parte das vezes não faz parte da equipa do projecto. Nesta metodologia, o cliente não se limita a expressar a sua visão do que pretende, trabalhando mesmo em conjunto com a equipa designada para o desenvolvimento. É da sua responsabilidade escrever requisitos de funcionalidades e testes funcionais, bem como decidir o timing para satisfação de cada requisito ou definir a prioridade de implementação das várias funcionalidades identificadas.

“Tester”: Responsável pela escrita de testes funcionais em conjunto com o cliente, ajudando-o a expressar as suas intenções sobre a forma de testes. Executa esses testes, reporta os resultados e gere as ferramentas que os gerem. São por vezes os próprios programadores.

“Tracker”: Personalidade encarregada de reportar o funcionamento de todo o processo XP, traçando as estimativas de toda a equipa e reportando a precisão destas. É sua função traçar o progresso de cada iteração, avaliar se o objectivo é atingível ou se algumas modificações são necessárias ao projecto.

Treinador (“Coach”): Pessoa responsável pelo processo como um todo. Guia todos os elementos da equipa e é tipicamente o elemento com mais experiência da equipa de programadores. Funciona como uma espécie de barreira aos vários programadores, mantendo-os sempre no caminho certo.

Consultor: Membro externo que possui a sabedoria tecnológica necessária ao projecto, guiando a equipa. É comum a contratação de um consultor especialista em certas ferramentas que a equipa de desenvolvimento não domina de forma satisfatória, para codificação da aplicação ou simplesmente para servir como guia de aprendizagem para a equipa de desenvolvimento.

Gestor (“Manager”): Elemento responsável por tomar decisões, que comunica com a equipa de projecto para determinar situação corrente e para perceber dificuldades e deficiências no processo e seguir rumos no processo de implementação do mesmo.

Práticas, Adopções e Experiências XP:

Tendo sido já referidas as várias práticas utilizadas neste tipo de metodologia de desenvolvimento, aquando da introdução do método de Programação Extrema, faço de seguida um pequeno acrescento relevante relacionado com o processo de desenvolvimento do XP:

- Este método é tipicamente marcado pelo facto do cliente tomar decisões de negócio enquanto os programadores se encarregam de decisões técnicas.
- O uso de casos de uso como forma de guiar todo o processo de exploração e definição de funcionalidades, representa uma prática simples e comum nesta metodologia de desenvolvimento. Esta é aliás a forma mais simples de tradução de conceitos em funcionalidades, sendo relativamente simples para clientes não familiarizados com processos informáticos, descreverem os objectivos da aplicação sobre a forma de esquemas como estes, e permitindo uma relação directa com classes a implementar no código aplicacional pelos programadores. Os casos de uso permitem fazer a ponte entre a linguagem comum e a linguagem programática e são por essa razão uma das bases mais adoptadas em XP.
- As equipas de desenvolvimento neste tipo de metodologia são formadas tipicamente por um número de pessoas compreendido entre 2 e 12 individualidades.

Em termos de adopção interessa referir os seguintes pontos:

- Sendo este método, um dos mais adoptados por empresas ao nível global, é por isso um dos mais fundamentados, e naturalmente um dos mais adoptados por empresas de tamanho pequeno/médio.

- O principal obstáculo ao sucesso da metodologia de Programação Extrema é a existência de qualquer resistência às suas práticas, por mais reduzida que esta seja. Qualquer factor que impeça uma comunicação, ou coordenação ideal entre os vários elementos da equipa é suficiente para levar ao insucesso de toda a metodologia e naturalmente do projecto em causa.

- A adopção de tecnologia incapaz de suportar mudanças ao nível de requisitos/especificações ou funcionalidades bem como de ferramentas que exijam muito tempo para retorno de feedback, não contribuem para o sucesso desta metodologia.

Em termos de relatos de experiência de uso do método, interessa referir:

- Um dos principais problemas do XP é, o facto de ser muito difícil estimar a duração das várias tarefas, por isso o uso de "spikes" - pequenas experiências em código que representam a forma de como um determinado problema pode ser mitigado – é uma forma de melhorar essa estimativa e de reduzir erros de implementação.

- O uso desta metodologia resulta no aumento do número de linhas de código produzidas, bem como do número de novos métodos e classes implementadas ao nível do código aplicacional.

Probabilidades de Aceitação no CI:

A melhor forma de perceber a aceitação da metodologia de programação extrema é enumerar os seus pontos positivos e implementáveis no Centro de Informática e os pontos que impedem a sua adopção por serem impraticáveis neste grupo de desenvolvimento:

(+) Desenvolvimento centrado em funcionalidades e pequenas iterações, acompanhado por testes e verificação de funcionalidades desenvolvidas, permite ter uma ideia constante do sucesso do desenvolvimento bem como do rumo a seguir em todo o processo.

(+) Papel do cliente na definição das várias funcionalidades, bem como o seu papel na definição dos vários protótipos da aplicação, permite que a satisfação dos requisitos seja o principal objectivo da aplicação, e permite um total envolvimento do cliente no desenvolvimento e com os elementos da equipa.

(+) A existência de uma fase no processo de desenvolvimento dedicada apenas à documentação do trabalho efectuado, é importante para o Centro de Informática, uma vez que permite que qualquer serviço desenvolvido fique devidamente documentado e se tenha no futuro uma ideia totalmente exacta do que foi feito e de como todo o processo foi guiado.

(+) Existência de papéis de responsabilidades relativamente simples e comuns, permite uma identificação exacta com a equipa de desenvolvimento existente no CI-FCUL, e permite uma distribuição rápida e objectiva dos diferentes papéis pelas várias individualidades.

(+) Semana de 40 horas de trabalho reflecte de facto a estrutura existente no Centro de Informática.

(+) A implementação de standards de código na implementação por parte dos programadores, relaciona-se de forma perfeita com a adopção de novas ferramentas de desenvolvimento por parte do Centro de Informática, uma vez que implementação baseada nestas, força exactamente a existência de certas regras aquando da implementação de cada nova aplicação. Desde regras relacionadas com aspectos estruturais de cada aplicação, a aspectos relacionados com a partilha de código pelos diferentes programadores.

(-) Total dependência do cliente pode tornar impraticável o desenvolvimento de uma aplicação, se este não tiver uma ideia exacta do que pretende ou não souber expressar devidamente as suas intenções. Tendo em conta que isso é comum no meio académico, deverá haver sempre uma moderação relativamente ao papel "supremo" do cliente.

(-) Técnica de programação em pares ("pair programming") baseada na programação levada a cabo por um "núcleo central" em torno de qual se situam todos os outros programadores, é impraticável no Centro de Informática, devido à existência de uma equipa de desenvolvimento muito pequena e devido à grande dependência que este tipo de técnica exigiria entre os vários programadores, levando ao esgotamento de recursos técnicos num único projecto. Impossibilitaria o desenvolvimento simultâneo de várias aplicações.

Conclusão

É então perceptível que o método XP é um método que apresenta um grande conjunto de características ideais de adopção no CI-FCUL, e que com alguns ajustes no seu funcionamento poder-se-á tornar a metodologia ideal para adopção por parte deste grupo de desenvolvimento.

SCRUM

"Chutar uma bola fora de campo de volta para o jogo" ***[Schwaber e Beedle, 2002]***

A declaração acima procura resumir a uma frase alegórica, a ideologia do método ágil de desenvolvimento conhecido como SCRUM. Uma metodologia que representa uma aproximação empírica ao desenvolvimento de sistemas, relacionada com controlo industrial de processos, resultando numa reintrodução de ideias flexíveis, adaptáveis e com especial foco na produtividade. O SCRUM concentra-se na forma como os elementos da equipa devem funcionar em conjunto de forma a produzirem um sistema flexível capaz de resistir e ser eficiente num ambiente em constante mudança.

Processo SCRUM:

O processo de desenvolvimento definido pela metodologia SCRUM envolve 3 fases: Pré-Jogo, Jogo e Pós-Jogo (Figura 51): ***[Schwaber, 1995; Schwaber e Beedle, 2002]***

Pré-Jogo: Fase inicial do processo que compreende todas as tarefas realizadas antes da implementação propriamente dita e que inclui duas outras subfases, a de Planeamento e a de Arquitectura/Desenho.

Planeamento: Nesta fase elabora-se a definição de todo o sistema a ser desenvolvido e suas funcionalidades, através da produção de uma "Backlog List" que contém todos os requisitos/funcionalidades conhecidos até ao momento, originários do cliente (como no XP), das divisões várias da empresa ou mais especificamente dos desenvolvedores. Os requisitos são prioritizados e é estimado o esforço para a sua implementação. No planeamento pré-jogo é ainda definida e identificada toda a equipa que fará parte do projecto, são equacionadas e adoptadas as ferramentas que permitem uma melhor concretização dos objectivos, é executado um levantamento dos riscos prováveis de ocorrerem em todo o processo de desenvolvimento, são elaboradas as várias formas de controlo do projecto e são identificadas as necessidades de processos de treino em ferramentas ou linguagens de implementação, em técnicas de comunicação, em formas de produtividade, etc. A cada iteração esta "Backlog List" é revista pelos membros da equipa e aprovada para a próxima fase.

Arquitectura: Partindo da "Backlog List" produzida na subfase de planeamento descrita anteriormente, é planeado o desenho de alto-nível do sistema, definindo-se a arquitectura a implementar no sistema. São ainda identificadas as mudanças necessárias à implementação dos requisitos, juntamente com os problemas que poderão vir a ser causados por essas mesmas mudanças. Para que o desenho final da aplicação possa ser aprovado, é essencial a realização de uma reunião final de revisão.

Desenvolvimento (Jogo): Fase de desenvolvimento do projecto propriamente dita. Constitui-se como a parte ágil de todo processo, sendo esta fase tratada à semelhança de uma caixa-preta de avião, em que é esperado o imprevisível e em que se procura dar resposta às mudanças efectuadas no sistema. As variáveis de ambiente, bem como as variáveis técnicas, são assim observadas e controladas através de diversas práticas. Todo o sistema é desenvolvimento em "Sprints", isto é, ciclos iterativos onde cada funcionalidade é desenvolvida de forma a acrescentar algo mais à aplicação.

Sprint: Subdivide-se em subfases distintas, são elas: Fase de requisitos, Análise, Desenho, Evolução e Lançamento. Toda a arquitectura e desenho do sistema são implementadas realmente durante estas subfases de Jogo, e cada uma dura normalmente entre 1 semana e 1 mês.

Pós-Jogo: Como o próprio nome indica, esta fase compreende tudo aquilo que se processa após a implementação das várias funcionalidades do sistema, funcionando assim como uma fase de conclusão das versões do produto. Esta fase é então apenas atingida sempre que existe um acordo para conclusão

das variáveis técnicas e de todos os requisitos identificados inicialmente. O Pós-Jogo inclui subtarefas relacionadas com a Integração do produto, com os testes finais e com a elaboração de toda a documentação relativa ao projecto em conclusão.

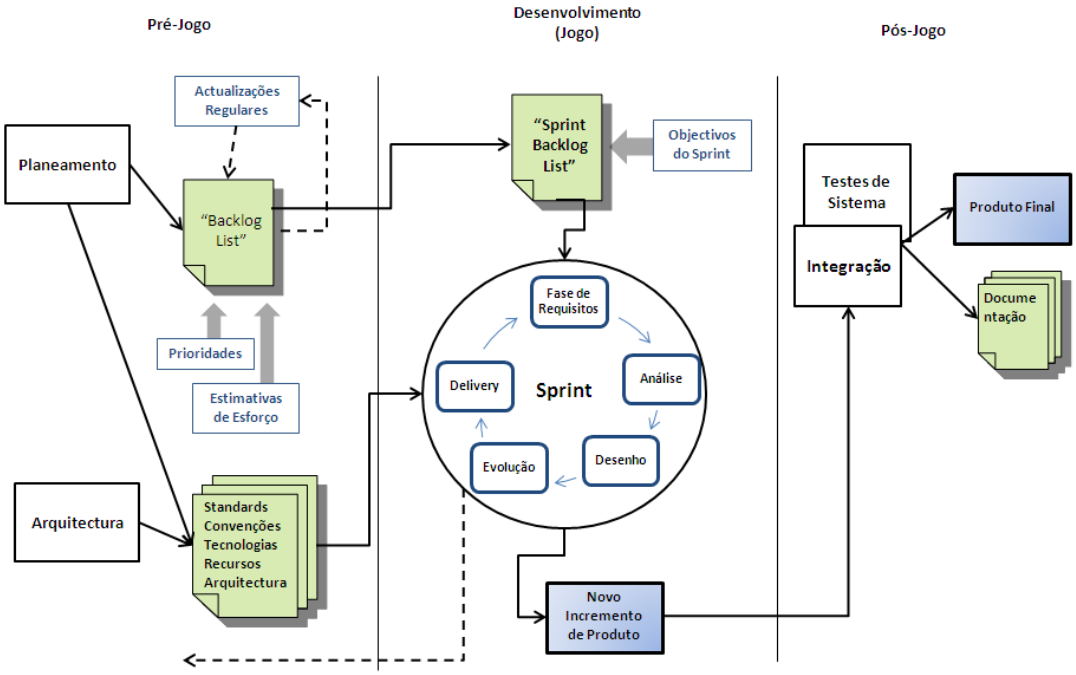


Figura 51: Esquema do Processo de Desenvolvimento de SCRUM

Papéis e Responsabilidades SCRUM:

Definem-se os seguintes papéis sobre os elementos da equipa de desenvolvimento SCRUM:

"Scrum Master": Responsável por assegurar que o projecto é levado a bom porto, segundo as práticas, valores, papéis e regras definidas pela metodologia. Faz com que o projecto possa progredir como planeado.

"Product Owner" (Proprietário do Produto): É o responsável oficial pelo projecto, sua gestão, e controlo, e de tornar visível a "Backlog List" com as especificações do produto. Tem o poder de tomar as decisões finais nas tarefas relacionadas com essa lista e participa na estimação da duração do projecto. É da sua responsabilidade transformar entradas de "Backlog" em funcionalidades a serem desenvolvidas.

Cliente: Existente em todos as metodologias ágeis, mas como papéis distintos em cada uma destas. No SCRUM o cliente participa nas tarefas relacionadas com a construção da lista de items do projecto, bem como em tarefas de decisões executadas sobre esta.

Gestor ("Manager"): Tem o papel de tomar decisões. Para além de participar directamente na definição dos objectivos e dos requisitos do projecto, ainda é sua função seleccionar o "Product Owner", impulsionar o progresso do projecto e reduzir a lista de "Backlog". Para que tudo isto seja possível, os

seus veículos de sucesso serão sempre os membros da sua equipa. O Gestor sozinho não cumpre qualquer uma das responsabilidades sobre si impostas.

Programador: Implementam as funcionalidades propriamente ditas, contribuem para a especificação inicial dos requisitos e funcionalidades do sistema e recebem ordens tal como reportam o trabalho desenvolvido ao Proprietário do Produto

Práticas, Adopções e Experiências SCRUM:

SCRUM é uma metodologia que não necessita, exige ou fornece qualquer prática específica no seu desenvolvimento, a não ser as que estão directamente relacionados com a composição do seu processo para desenvolvimento de produtos. No entanto, e como forma de evitar o caos provocado pela imprevisibilidade e complexidade de todo o método de trabalho, o SCRUM necessita de várias práticas de gestão ou ferramentas em várias fases do desenvolvimento. A descrição das práticas em seguida explicadas relaciona-se então exclusivamente com práticas directamente relacionadas com o ciclo de vida de produção de uma produto quando é usada a metodologia SCRUM: ***[Schwaber e Beedle, 2002]***

- **Construção de um Backlog para o Produto**: A manutenção de uma lista constantemente prioritizada de requisitos a serem inseridos no sistema e é condição obrigatória no uso desta metodologia. Esta lista tanto pode ser construída apenas pelo cliente (em casos em que este sabe especificamente o que quer e tem uma ideia da forma de implementar o que pretende) como pode ser produto da colaboração do cliente com a equipa de desenvolvimento, como ainda pode resultar da experiência da equipa com breves consultas das intenções do cliente do produto. Em qualquer um dos casos, a construção da "lista de Backlog" baseia-se sempre na tecnologia correntemente ao alcance da equipa.

- **Desenvolvimento por Sprints:** A prática que melhor define todo o desenvolvimento apoiado nesta metodologia é a de capacidade de adaptação constante a ambientes em permanente mudança. Esses procedimentos de adaptação a tempo, requisitos, recursos e tecnologia traduzem-se na produção de uma aplicação no fim de cada interacção, com características que respondem eficazmente às várias alterações observadas. Assim, existem 4 práticas comuns durante períodos de Sprint:

    - **Reuniões de Planeamento**: Decidem objectivos e funcionalidades.

    - **Reuniões de Backlog:** Seleccionam itens para "Backlog" e para desenvolvimento no produto.

    - **Reuniões Diárias:** Gerem o progresso de todo o desenvolvimento.

    - **Reuniões de Revisão**: Apresentam resultados do desenvolvimento e projectam próximo Sprint.

Todas estas actividades de gestão e reuniões frequentes são por isso designadas por – "Scrums" – e permitem identificar e corrigir deficiências ou obstáculos ao desenvolvimento.

No que se refere a pormenores de adopção e a relatos de experiência interessa referir os seguintes pontos:

- Como já foi dito, uma das características verificadas no estudo da metodologia ágil SCRUM é precisamente o facto de não necessitar de qualquer prática específica de engenharia e de apenas se apoiar num conjunto de práticas relacionado com controlo e gestão de desenvolvimento. Por esta razão esta metodologia torna possível a integração de práticas de engenharia conhecidas sem que seja gerado qualquer conflito com as suas exigências.

- Outra característica verificada após o uso da metodologia por diversas organizações, é o facto de os vários papéis a desempenhar dentro da equipa de trabalho poderem mudar constantemente, e serem misturadas ou alteradas durante o processo. O gestor do projecto pode por alguma razão ser mudado a meio do projecto, para que haja uma melhor adaptação a uma qualquer alteração ambiental verificada, ou simplesmente para tentar corrigir algum insucesso momentâneo do desenvolvimento.

- O SCRUM pode ser tipicamente adoptado em duas situações de desenvolvimento: em projectos já existentes, sempre que nos encontramos numa situação em que o ambiente e as ferramentas de desenvolvimento já existem mas a equipa apresenta problemas em lidar com a mudança ou a complexidade; ou no desenvolvimento de um novo projecto, trabalhando com a equipa e o cliente por vários dias para construir um "Backlog" de produto inicial.

- De experiências do uso da metodologia foi possível concluir também que esta não é feita para ser usada por grandes ou complexas estruturas de equipas, estas podem, no entanto, aproveitar certas fases desta metodologia, ou práticas relacionadas com a mesma, quando se encontram isoladas em grandes projectos. Uma equipa pequena formada por menos de 10 elementos e por pelo menos 5 pessoas é a constituição ideal para um processo de desenvolvimento como este.

- Por fim verifica-se que a adopção e uso de SCRUM provoca um aumento de voluntarismo dentro da equipa, e leva a uma mais eficiente resolução de problemas.

Probabilidades de Aceitação no CI:

A melhor forma de perceber a aceitação da metodologia SCRUM no CI-FCUL é enumerar os seus pontos positivos e implementáveis no Centro de Informática e os pontos negativos da sua adopção por parte desta instituição:

(+) A forma como a metodologia SCRUM não se prende a grandes exigências e a forma como é baseada num desenvolvimento centrado em funcionalidades garante-lhe grande elasticidade aquando da mudança de requisitos ou de conceitos de utilização no desenvolvimento.

(+) A forma controlada como todo o processo é guiado através de grande número de reuniões entre os envolvidos, bem como através da definição dos vários Sprints de desenvolvimento, permitem uma não estagnação no desenvolvimento de funcionalidades, e evitam grande número de erros de interpretação.

(-) O facto de SCRUM não guardar no seu processo de desenvolvimento qualquer fase (por mais curta que seja) para a escrita de documentação referente ao projecto em desenvolvimento, faz com que no final da implementação deste, não se tenha qualquer ideia da "história" do seu desenvolvimento, e impossibilita o entendimento do desenvolvimento do projecto no futuro por parte de desenvolvedores que não estiveram envolvidos no processo.

(-) O grande número de reuniões exigidas no processo de desenvolvimento, podem acabar por ocupar grande parte do tempo de desenvolvimento, o que em equipas desorganizadas e sem ideias bem definidas do que se quer no projecto, pode significar o rotundo falhanço do desenvolvimento, ou um grande atraso no mesmo.

(-) A metodologia SCRUM deixa por definir vários pormenores importantes numa metodologia deste género, como por exemplo o número de horas de trabalho adequadas, o uso ou não de ferramentas ou a implementação de padrões de programação

(-) A principal falha ao nível desta metodologia é no entanto, o facto de esta não definir qualquer forma de recolha de requisitos ou de arranque de um projecto (como acontece de forma clara na metodologia de desenvolvimento de programação extrema), o que resulta numa grande incerteza em todo o funcionamento da metodologia. Ao não se saber técnicas exactas de apurar as funcionalidades necessárias a um projecto, e de partir para o início de desenvolvimento deste, fica de sobremaneira dificultado o entendimento entre clientes e desenvolvedores, uma vez que os seus níveis de conhecimento técnico são bastante diferentes.

(-) Por fim, a obsessão pela construção e manutenção de uma lista de funcionalidades prioritizada pode levar à existência de uma "visão em túnel" no projecto, e a uma incapacidade de conclusão do mesmo por essa lista de funcionalidades nunca chegar a ficar vazia. Isto ocorre principalmente, se não houver uma forma clara de entendimento das funcionalidades entre os desenvolvedores e o cliente. "Scrum won't fix your culture, will expose it".

Conclusão

Através da constatação do grande número de pontos negativos identificados, facilmente se percebe que esta metodologia não é adequada para utilização no Centro de Informática da FCUL, sendo demasiado subjectiva a sua interpretação para que desenvolvimentos sobre a sua filosofia resultem nos resultados esperados.

Metodologias Crystal Family

"Cristal Family", tal como o próprio nome o sugere, é um conjunto de metodologias relativamente parecidas (pertencem por isso a uma família) em que a selecção da melhor metodologia é feita de acordo com cada projecto em desenvolvimento, ou com cada tarefa específica de um projecto, associando metodologias de diferentes cores (e por isso diferentes pesos) – quanto mais escura a cor, mais pesada a metodologia – a diferentes graus de complexidade de desenvolvimento. Assim, a escolha da cor apropriada, baseia-se na criticalidade e na extensão de cada projecto, tendo-se sempre em conta que para projectos maiores, existe uma necessidade naturalmente maior de coordenação no desenvolvimento e por essa razão é necessária uma metodologia mais pesada.

Em qualquer uma das metodologias pertencentes a esta família existe sempre um ciclo de desenvolvimento incremental (típico aliás, das metodologias ágeis), dando-se especial ênfase à comunicação e à cooperação entre elementos de equipa. Existe também um esforço para tentativa de redução do número de versões intermédias, antes da conclusão do projecto. No entanto, todos estes objectivos não exigem grandes práticas específicas de desenvolvimento ou ferramentas restritas de implementação, permitindo por essa razão a adopção de outras metodologias para trabalho simultâneo.

Falando-se então mais especificamente na classificação das várias metodologias "Crystal Family", temos que segundo o tamanho da equipa de desenvolvimento estas podem ser classificadas da seguinte forma:

"Clear" (Branca): Equipa de desenvolvimento muito pequena (6 pessoas)

"Yellow" (Amarela): Equipa de desenvolvimento pequena (20 pessoas)

"Orange" (Cor-de-Laranja): Equipa de desenvolvimento média (40 pessoas)

"Red" (Vermelho): Equipa de desenvolvimento grande (80 pessoas)

Destas 4 metodologias, destacam-se principalmente duas ***[Cockburn, 2002]*:**

- **"Crystal Clear":** A pequena equipa de desenvolvimento localiza-se num espaço partilhado aberto como forma de contornar as limitações na estrutura de comunicação. Em média o tempo de implementação de todo um projecto, estende-se por apenas 8 semanas. Esta é a única das metodologias que valerá a pena considerar a adopção no âmbito deste projecto uma vez que as restantes dizem respeito a projectos de maior escala por equipas mais extensas.

- **"Crystal Orange":** É ao contrário da anterior, desenhada para projectos de média dimensão, contanto por isso com uma equipa de desenvolvimento já relativamente extensa. Conta também, devido à sua maior complexidade, com uma duração de desenvolvimento do projecto entre 1 e 2 anos. Anos esses em que o projecto é dividido pelas diversas equipas, assumindo estas, funções diversas, cruzadas e por isso alteráveis. É dada especial atenção ao tempo de lançamento do novo produto no mercado, analisando-se os tipos de contrapartidas possíveis entre tempo de desenvolvimento e a qualidade do produto lançado, sabendo-se que a rápida mudança nos

requisitos e no desenho do produto poderá resultar na redução dos custos de manutenção, permitindo que a eficiência seja mantida.

Processos Crystal Family:

Os processos de desenvolvimento das metodologias pertencentes a esta família são, regra geral, formados por 2 fases principais (Figura 52):

"Staging": Processo de recolha e construção de requisitos, e que também inclui o planeamento da próxima versão e calendarização das várias tarefas a executar em cada iteração.

Iterações: Desenvolvimento de funcionalidades feito de forma paralela e sempre de acordo com os vários standards de implementação adoptados. Monitorização de progresso é comum tanto como a construção de demonstrações e lançamentos de protótipos funcionais.

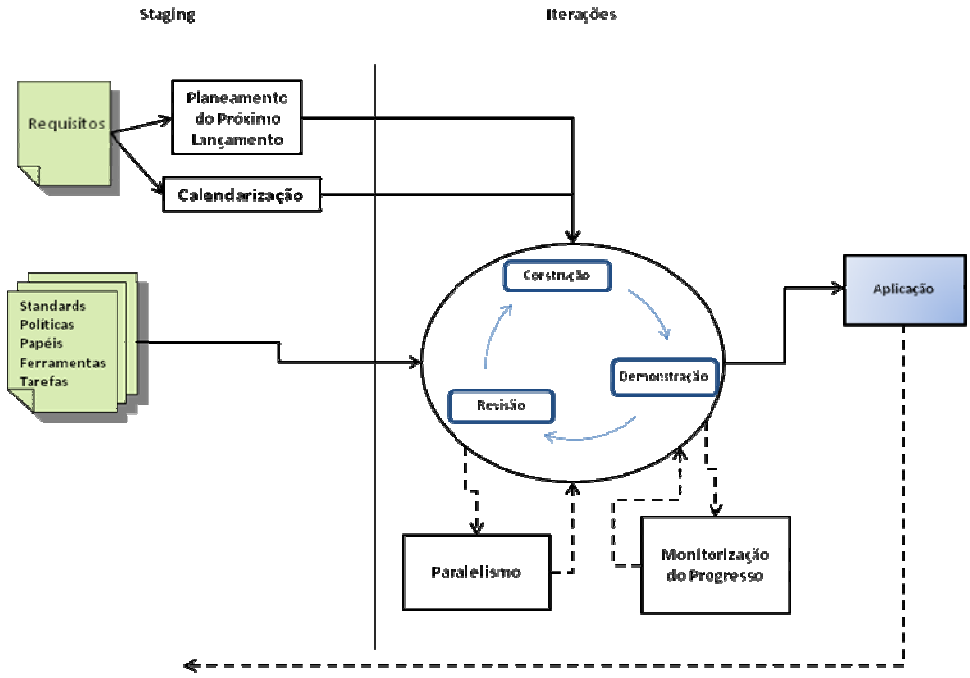


Figura 52: Esquema do Processo de Desenvolvimento de Crystal Family

Papéis e Responsabilidades Crystal Family:

A grande diferença entre as duas metodologias em termos de papéis e responsabilidades que são atribuídas e/ou distribuídas pelos vários elementos das equipas de desenvolvimento é, o facto de na metodologia "Crystal Clear" apenas existir uma equipa para um projecto, enquanto em "Crystal Orange" existe multiplicidade de equipas num único projecto. Em ambas, no entanto, um tipo de trabalho, tarefa ou responsabilidade, pode compreender vários papéis e ser desempalhado por várias pessoas diferentes ***[Cockburn, 2002]***:

Especificidades de "Crystal Clear":

No método a que devo mais atenção, o método Crystal Clear, os papéis principais exigem a existência de individualidades diferentes para desepenharem essas funções. Esses papéis de principal importância são:

Patrocinador – É o responsável por reunir condições necessárias para suporte ou sustento do projecto;

Programador-Desenhador Sénior – Mais experiente que um simples desenhador ou programador, e responsável pela integração dos restantes, ou pelo bom funcionamento de uma inteira equipa de programadores ou desenhadores;

Utilizador – Desempenha papel importante na descrição de requisitos ou funcionalidades, e participa em períodos de teste e na verificação de funcionalidades. Estes papéis, dividem-se em papéis mais específicos como **Coordenador de Projecto**, **Expert de Negócio** ou **Angariador de Requisitos.**

Especificidades de "Crystal Orange":

Para além dos papéis típicos de uma metodologia ágil e para além dos métodos adicionais descritos acima na metodologia "Crystal Clear", o outro método mais relevante desta família – "Crystal Orange" – aumenta complexidade em todo o processo, sugerindo mais alguns papéis importantes e agrupando-os em equipas funcionais com diferentes papéis e que entram em acção em diferentes fases do projecto. Integram estas equipas os seguintes papéis: **Desenhador UI**, **Desenhador BD**, **Perito de usabilidade**, **Facilitador técnico**, **Analista**, **Arquitecto**, **Escritor** e "**Tester"**. Todas a especificações destas tarefas mais estritas no desenvolvimento de projectos, permitem não só reduzir o número de versões de um produto como também promover uma comunicação mais eficaz dentro da equipa de desenvolvimento enquanto se regista um aumento da qualidade do produto. Naturalmente que para que seja possível tanta envolvência num projecto, são necessárias muitas mais individualidades que no método mais simples desta família de metodologias, e daí a diferença entre 6 ("Crystal Clear") e 40 ("Crystal Orange") pessoas.

Práticas, Adopções e Experiências Crystal Family

Especificando mais o típico processo de desenvolvimento baseado em "staging" e iterações, para práticas directamente relacionadas com o desenvolvimento incremental de projectos, conseguimos escrutinar algumas práticas como ***[Cockburn, 1998]***:

- Realização de **Reuniões de "Staging":** Planeamento do próximo incremento do sistema, bem como de todos os requisitos e funcionalidades que serão satisfeitos por este.

- **Provas de Revisão**: Cada incremento inclui várias iterações e cada iteração inclui actividades de construção, demonstração e revisão dos objectivos do incremento. As provas de funcionalidades

permitem verificar o funcionamento correcto bem como desencadear correcções em futuras iterações.

- **Monitorização**: A medição contínua do progresso do desenvolvimento é das actividades mais comuns durante todo o processo, tanto através de provas de revisão como também através da instituição de pontos de referência e da monitorização directa de toda estabilidade dos processos de implementação.

- **Implementação Paralela e Fluxo Contínuo de Desenvolvimento**: É prática característica desta família de metodologias, assim que uma tarefa é dada como estável, uma próxima tarefa arrancar. No caso do "Crystal Orange", a existência de várias equipas de desenvolvimento permite aumentar ainda mais a velocidade de desenvolvimento fazendo uso paralelo destas, realizando-se tarefas em simultâneo, sempre que estas não desencadeiam conflitos entre si.

- **Modificação de Tecnologia:** Trabalhar e alterar uma tecnologia centrando-a nas especificidades do projecto, permite melhorar o seu processo de desenvolvimento. "Workshops" no início ou no fim do desenvolvimento de uma funcionalidade, são a forma mais comum de isto ser feito.

- **Vistas de Utilizadores**: Manter duas (2) visões de utilizador (ou conceitos de utilização) por cada nova versão, é uma forma comum de encontrar falhas, ou possibilitar diferentes tratamentos de informação.

- **Reuniões de Reflexão:** Realizam-se após e anteriormente às fases de incrementação e entre os vários elementos de equipa, com objectivo de chegar a conclusões sobre processos, possibilitando a evolução em desenvolvimentos ou iterações futuras.

- Aliado às práticas já identificadas, existem típicos produtos de trabalho utilizados ou produzidos pelos métodos "Crystal Clear" e "Crystal Orange":

    - Ambas incluem: **Modelos Sequenciais de Lançamento**, **Modelos de Objectos**, **Manual de Utilizador**, **Testes** e **Migração de Código**.

    - "Crystal Clear" inclui ainda C**asos de Uso anotados** e **Descrições de Funcionalidades**, **Rascunhos de ecrãs** e **Notas de desenho,**

    - "Crystal Orange" requer **Documentação de Requisitos** e **Documentação do Desenho**.

- Por fim, existem ainda algumas ferramentas usadas por estes dois principais representantes desta família de metodologias:

    - **"Crystal Clear"** requer um **compilador,** uma **ferramenta de gestão e configuração de versões**, e utiliza **quadros brancos** para representação e anotação, evitando assim outro tipo de documentação.

- **"Crystal Orange"** requer **ferramentas de gestão de versões e código**, **ferramentas de testes**, **ferramentas de comunicação**, **ferramentas de medição da evolução do projecto e de medição de performances**.

- "Crystal Orange" propõe ainda o **uso e selecção de standards** para anotação, desenho, formatação e qualidade.

Falar de pormenores e problemas verificados durante a adopção e experimentação de metodologias desta família, é sempre complicado, uma vez que tipicamente, para metodologias com reduzido número de elementos na equipa de desenvolvimento dão-se certo tipo de ocorrências, enquanto para metodologias que usufruem de várias equipas de desenvolvimento e paralelismo na implementação, os problemas e factos de utilização são de um género completamente diferente. De qualquer forma:

- Os principais problemas emergentes na adopção de uma metodologia pertencente à família de métodos Crystal Family são: a deficiente comunicação dentro e entre equipas no desenvolvimento, uma fase de recolha de requisitos demasiado extensa - provocando que o não desenho de uma arquitectura se estenda por muitos meses -, e papéis mal definidos e mal distribuídos pelos membros que constituem a equipa de desenvolvimento.

- Descobriu-se que a adopção de um processo iterativo distinto para cada tipo de incremento no sistema (para cada tipo de funcionalidade por exemplo) permitiu ultrapassar as dificuldades conduzindo a uma maior ligação entre elementos de equipa. A clarificação de papéis de cada pessoa envolvida, o planeamento de linhas de comunicação bem como uma gestão de suporte intensa, permitiram não só aumentar a eficiências dos processos, como também fizeram com que as metodologias mais simples desta família caíssem em desuso, por conseguinte, impulsionaram um uso sustentado das metodologias mais complexas.

- Uma particularidade importante no uso destas metodologias, é a existência de individualidades que se destacam pelos seus conhecimentos aprofundados de uma certa ferramenta, de costumes da empresa, ou das próprias formas de trabalho da metodologia adoptada. A existência destes "trunfos", constitui uma forma de impulsionar os restantes elementos e de garantir que certas tarefas são sempre realizadas, forçando o sucesso do projecto em desenvolvimento.

- Concluindo sobre o uso da metodologia "Crystal Clear", resume-se o seu uso adequado a apenas uma única equipa, numa simples sala de trabalho, e o seu uso exclusivo para projectos não críticos, devido à escassez de elementos de validação e avaliação do sistema.

- Sobre a metodologia mais complexa, "Crystal Orange", é de sumarizar que esta requer a existência de várias equipas de desenvolvimento localizadas num único edifício, sendo crítica a capacidade de estruturação de subequipas e a eficiência nas actividades de verificação de código e funcionalidades. A existência de um maior número de papéis e responsabilidades permite dar resposta a esta maior exigência da metodologia.

- Por fim, finalizo esta abordagem referindo que qualquer um dos membros desta família de métodos ágeis de desenvolvimento, não limita as práticas de desenvolvimento apenas às que o caracterizam, podendo por isso adoptar práticas de outras metodologias (como a de Programação Extrema – XP), adoptando produtos de trabalho como: casos de uso, ecrãs rascunho e sketches de desenho.

Probabilidades de Aceitação no CI:

A melhor forma de perceber as hipóteses de adopção e aceitação desta família de metodologias no CI-FCUL é enumerar os seus pontos positivos e negativos para o Centro de Informática:

(+) Uma das grandes vantagens deste tipo de metodologias ser expresso numa inteira família de processos, é o facto de ser possível, numa organização com diversos tipos e dimensões de projectos, adoptar tarefas que se adaptam a cada género diferente. Desta forma, não existem projectos fora do alcance desta família de metodologias, podendo a mesma equipa ser responsável pelo desenvolvimento dos mais simples e dos mais complexos projectos.

(+) Na metodologia "Crystal Orange", é possível a implementação simultânea de vários projectos, através da divisão da equipa de desenvolvimento em várias subequipas. Esta possibilidade permitiria uma aceleração dos tempos de desenvolvimento dos projectos, permitindo uma resposta mais rápida por parte do CI-FCUL.

(-) No entanto, contrapondo os dois pontos anteriores, no CI-FCUL não existe grande variedade de projectos, possuindo todos, consideravelmente o mesmo tipo de complexidade, e sendo todos desenvolvidos pelo mesmo tipo de equipa de desenvolvimento. Não existe por isso necessidade de adopção de uma família de metodologias deste género, mas apenas de uma simples metodologia aplicável a todos os projectos.

(-) No âmbito de adopção de apenas a metodologia "Crystal Clear" (a única possível de implementar numa equipa de desenvolvimento como a do CI-FCUL), existem vários factores em desfavor desta metodologia, que passo de seguida a enumerar:

(-) Antes de mais, a falta de especificação de formas de recolher requisitos e de arrancar com qualquer projecto, conduzem a uma fase de requisitos demasiado longa e desviada dos problemas e funcionalidades principais de implementação.

(-) Em segundo lugar, a inexistência de uma fase de revisão e de testes de funcionalidades no processo de desenvolvimento "Crystal Clear", torna o desenvolvimento altamente propenso a erros e conduz a aplicações instáveis.

(-) Em terceiro lugar, devido à falta de mecanismos e papéis de gestão de projectos em desenvolvimento e da própria equipa, resulta em grandes dificuldades de comunicação interna entre os vários elementos, levando mais uma vez a um processo de desenvolvimento demasiado arriscado.

(-) Essa falta de comunicação, e a tentativa de a corrigir leva a uma maior especificação de papéis de responsabilidades, necessitando de um aumento da equipa de desenvolvimento, aumento esse, que é não é possível de implementar correntemente no órgão em causa.

(-) Por fim, a documentação limitada à discussão e escrita de pormenores técnicos num quadro branco em detrimento de um total relato do processo de implementação das aplicações leva a um desconhecimento futuro sobre o projecto, não permitindo a desenvolvedores não envolvidos, perceberem pormenores importantes da sua concretização.

Conclusão

O grande número de pontos negativos, e portanto de problemas, levantados na adopção de uma metodologia desta família deixa antever a sua impossibilidade de adopção no CI-FCUL, perdendo claramente para metodologias melhor definidas e mais centradas nos tipos de projecto existentes neste órgão de desenvolvimento.

Desenvolvimento Orientado a Funcionalidades (FDD)

O Desenvolvimento Orientado a Funcionalidades (ou em inglês, "Feature Driven Development") é uma aproximação ágil e adaptativa de todo o processo de desenvolvimento de software, que se foca com especial atenção nas fases de desenho e implementação. Uma vez verificada uma maior eficiência por parte deste tipo de metodologias, o FDD também adopta um desenvolvimento iterativo ao nível dos seus processos. Assim, existe uma especial preocupação com aspectos de qualidade e a construção de versões frequentes e palpáveis, bem como a monitorização apurada do progresso de todo o projecto, são aspectos fundamentais da metodologia.

O FDD consiste basicamente em cinco processos sequenciais de desenvolvimento (descritos em pormenor mais à frente), e fornece os métodos, técnicas e guias necessários para que qualquer projecto seja levado a bom termo (ao contrário de grande parte das metodologias deste género, que são na generalidade pouco específicas ou mal definidas). Inclui ainda papéis, artefactos, objectivos e limites de tempo, necessários à realização de diversos tipos de projectos e é possível de usar no desenvolvimento de sistemas críticos ***[Palmer e Felsing, 2002].***

Processos FDD

Antes de descrever o processo de desenvolvimento FDD convém ter em mente que esta é uma metodologia orientada para a produção de resultados funcionais, frequentes e palpáveis, e por essa razão todas as tarefas que integram o processo de desenvolvimento devem suportar rápidas adaptações bem como alterações tardias nos requisitos.

Quanto à duração do processo que descrevo em seguida, uma iteração demora tipicamente entre 1 a 3 semanas a ser concluída, levando a um desenvolvimento com fases relativamente curtas e como o próprio nome indica, orientada para a implementação primeira de funcionalidades.

Temos assim 4 fases bem defininas (Figura 53) ***[Palmer e Felsing, 2002]***:

1. **Desenvolver um Modelo Geral:**
    - **a.** Este modelo, parte do princípio que aquando do início do desenvolvimento, todos se encontram conscientes da extensão, do contexto e sobretudo dos requisitos ou especificações, que se esperam no projecto.
    - **b.** Construção de um "guia" de entendimento do projecto é realizada por "Especialistas de domínio" para os restantes membros da equipa.
        - **i.** Domínio geral é dividido em diferentes áreas de domínio, e guias detalhados são feitos para cada uma, pelos seus membros.
    - **c.** Equipas de desenvolvimento trabalham em pequenos grupos para produzirem objectos para cada área de domínio (de acordo com os guias). Existindo apenas uma equipa de desenvolvimento, a coordenação é facilitada
    - **d.** Simultaneamente um modelo geral do sistema vai tomando forma.
2. **Construir uma lista de funcionalidades:**
    - **a.** Com base nos guias, nos modelos de objectos construídos e na documentação existente de requisitos, constrói-se uma lista de funcionalidades para o sistema a desenvolver.
        - **i.** Os conjuntos de funcionalidades maiores são divididos em conjuntos mais pequenos que representam as diferentes actividades nos diferentes domínios.
    - **b.** A lista de funcionalidades é revista por utilizadores, clientes e patrocinadores do sistema.
3. **Plano por funcionalidade:**
    - **a.** Criação de planos de alto nível onde o conjunto de funcionalidades são sequenciadas segundo prioritização adequada e atribuídas a programadores chefes (principais).
    - **b.** As classes identificadas no desenvolvimento de um modelo geral, são atribuídas a desenvolvedores individuais (denominados em inglês por "classe owners").
    - **c.** Calendarização e principais pontos de objectivo ("milestones"), são identificados para conjuntos de funcionalidades futuras que se desejam ver implementadas.
4. **Desenho e Construção de Funcionalidades:**
    - **a.** Conjunto pequeno de funcionalidades é seleccionado da lista de funcionalidades construída e são formadas equipas devidamente associadas a cada conjunto.
    - **b.** Cada funcionalidade é construída com base numa implementação iterativa demorando de poucos dias a 2 semanas. Processo iterativo inclui tarefas como: Inspecção de Desenho, Codificação, Testes de Unidades e Integração e Inspecção de Código.
    - **c.** Funcionalidades completas são integradas na base geral da aplicação.

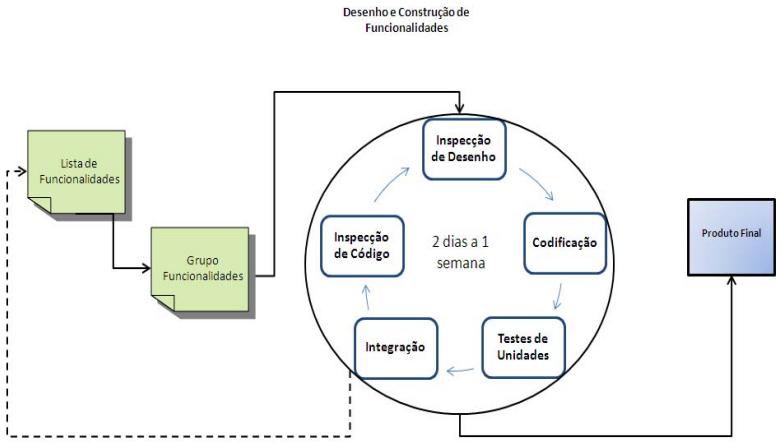


Figura 53: Esquema do Desenho e Construção de Funcionalidades, 4ºponto do processo de Desenvolvimento de FDD

Papéis e Responsabilidades FDD

Os papéis ou responsabilidades que uma metodologia orientada ao desenvolvimento de funcionalidades emprega sobre os seus membros, classificam-se em 3 categorias principais - papel principal, papel secundário e papel adicional - e suas subcategorias. ***[Palmer e Felsing, 2002]*:**

Papéis Principais:

Gestor de Projecto: É o líder do projecto, aquele que acima de tudo protege a equipa de desenvolvimento do meio exterior/circundante e fornece a estes as condições de trabalho necessárias ao bom desenvolvimento do projecto.

Arquitecto Chefe: Responsável pelo desenho geral do sistema conservando o poder de tomar as decisões finais em todas as matérias relacionadas com o desenho.

Gestor de Desenvolvimento: Coordena e lidera as actividades diárias de desenvolvimento e resolve os conflitos existentes no seio da equipa. É ainda responsável por resolver os problemas relacionados com recursos.

Programador Chefe: Toma parte na análise de requisitos e desenho dos projectos. Lidera pequenas equipas na análise, selecção, desenho e desenvolvimento de novas funcionalidades. É ele ainda que identifica ou atribui responsabilidades de "class owners", e a par com o gestor de desenvolvimento resolve problemas técnicos e de recursos.

"Class Owner" (Proprietário de Classe): Desenvolvedor que trabalha sob o programador chefe em tarefas de desenho, codificação, testes e documentação e que é responsável pelo desenvolvimento da classe que lhe é entregue.

"Domain Expert" (Perito no Domínio): Pode ser um utilizador, um cliente, um patrocinador, um analista ou uma mistura de todos estes. Detém a sabedoria de como implementar diferentes requisitos e passa essa sabedoria aos desenvolvedores.

Papéis Secundários:

Gestor de Versões: Responsável por controlar o progresso do processo, revendo os relatórios dos programadores, e realizando reuniões para monitorização de progresso, revisão de funcionalidades ou definição de novos objectivos.

Engenheiros de Montagem: Fazem a montagem, mantêm e correm todo o processo de integração de funcionalidades na aplicação principal.

"Toolsmith": Programador que constrói pequenas aplicações intermediárias para desenvolvimento ou teste.

Administrador do Sistema: Configura, gere e testa os servidores, a rede, e as várias estações de trabalho e ambientes de teste usados. Participa em toda a produção do sistema e é normalmente uma das individualidades com mais conhecimento na área de negócio da empresa.

Papéis Adicionais:

"Tester": Verifica que o sistema a ser produzido cumpre todos os requisitos do cliente.

"Deployer": Converte informação de variados formatos para formatos requeridos pelo sistema, participando no lançamento e produção de novas versões.

Escritor Técnico: Prepara toda a documentação do utilizador e tem conhecimentos gerais do negócio e das tecnologias usadas.

É de esclarecer que a extensa lista de papéis/responsabilidades apresentada identifica apenas papéis necessários de atribuir a individualidades mas não implica que todas elas sejam atribuídas a diferentes elementos. Um elemento de equipa pode desempenhar vários papéis, e num projecto muito curto, é comum 3 ou 4 pessoas chegarem mesmo a desempenhar todas estas funções. ***[Palmer e Felsing, 2002]*.**

Práticas, Adopções e Experiências FDD

Uma das questões que poderá ter sido levantada aquando da descrição do processo de desenvolvimento implementado por esta metodologia, é o facto de não ser descrito qualquer procedimento de recolha de requisitos ou a forma como é feita a "ponte" entre o levantamento de requisitos e a construção de uma lista de funcionalidades a implementar no projecto. No entanto, são práticas correntes da metodologia FDD:

- Usar e construir modelos UML (e por conseguinte pedaços de código funcional) no processo de recolha de requisitos. Estes modelos em UML permitem ter uma ideia do enquadramento geral dos

requisitos e das funcionalidades no sistema, enquanto permitem ao mesmo tempo um esclarecimento do número e do tipo de classes que serão necessários de implementar em termos programáticos.

- O uso de documentos textuais simples (para além destes diagramas em UML), que derivam tipicamente de casos de uso construídos com o auxílio do cliente do projecto e que tipicamente retratam funcionalidades que este deseja ver satisfeitas pelo sistema.

- Podem ainda ser definidos diagramas de classes e diagramas de sequência, associando funcionalidades (presentes em casos de uso) a classes e objectos (presentes nos documentos textuais e na "ideia" do cliente).

Todas as práticas em seguida listadas estão directamente relacionadas com o processo de desenvolvimento da metodologia bem como com os papéis desempenhados pelos membros da equipa de desenvolvimento, e podem existir em simultâneo como podem existir de forma individual no processo de desenvolvimento, dependendo logicamente da complexidade do projecto em mãos:

- **Modelação de Objectos de Domínio**: É essencial para o arranque de um projecto a exploração e a representação de um domínio exemplificativo do problema que origina a construção de um sistema. E esta modelação resulta numa plataforma onde as funcionalidades vão sendo adicionadas e integradas com a evolução na implementação desse domínio.

- **Desenvolvimento por Funcionalidade**: São típicos desta metodologia, um desenvolvimento baseado numa lista decomposta de funcionalidades, uma monitorização do progresso da implementação individual das funcionalidades identificadas e a existência de funções que permitem aos clientes validar as mesmas.

- **'Ownership' de cada Classe**: Cada classe tem uma pessoa responsável pela sua consistência, performance e integração conceptual, bem como equipas pequenas e dinamicamente formadas à volta deste.

- **Montagens Regulares**: Uma das práticas mais característica do sistema, bem como de todo o tipo de metodologias ágeis, é assegurar que existe sempre um sistema demonstrável em execução, e que novas funcionalidades vão sendo adicionadas assim que os devidos de processo de testes e validação são concluídos.

- **Gestão de Configurações:** A identificação e monitorização de versões de cada ficheiro de código concluído é essencial para que seja possível manter sempre uma versão da aplicação funcional, e para que adições à aplicação sejam sempre facilmente distinguíveis entre si.

De todas as práticas referidas para esta metodologia, a equipa de desenvolvimento pode adoptar as que considera adequadas ao seu nível de experiência e ao projecto em causa, não sendo por isso obrigatório o uso de todas estas metodologias, mas sim típico o uso de algumas.

Do vasto uso e da adopção experimental desta metodologia ágil de desenvolvimento, retém-se principalmente a sua polivalência para a execução tanto de novos projectos como para a actualização e adição de funcionalidades a projectos já em desenvolvimento. Devido a esta característica, a adopção do método de desenvolvimento orientado a funcionalidades permite o desenvolvimento e a entrega de sistemas de negócio críticos com a devida qualidade e no devido tempo. Esta mais-valia de utilização, torna o processo bastante apelativo para uso numa organização de médio/grande porte, mas no entanto, devido à forma bastante restritiva como o FFD define o seu processo de recolha de requisitos, bem como os seus processos de controlo de desenvolvimento, tornam-no difícil de implementar numa organização com poucos recursos.

Probabilidades de Aceitação no CI:

A melhor forma de entender a possibilidade de aceitação da metodologia FDD no CI-FCUL é através da enumeração dos seus pontos positivos, bem como dos factores que parecem limitar essa mesma adopção:

(+) O primeiro factor positivo nesta metodologia é o facto de ser completamente orientada para a obtenção de resultados, centrando-se no desenvolvimento de funcionalidades de forma prioritizada, mas contemplando alterações tardias nas especificações. No CI-FCUL esta é a mentalidade de desenvolvimento que se requer para um rápido e ágil desenvolvimento de serviços.

(+) Outro factor positivo, é o facto de ser estimável a apresentação de novas funcionalidades a cada semana de desenvolvimento, uma vez que cada iteração (que contempla no mínimo a implementação de uma funcionalidade) demora tipicamente uma semana. No CI-FCUL é importante a apresentação de resultados desde o início do desenvolvimento de um serviço, para que os clientes possam confirmar as suas expectativas e ideias relativamente ao mesmo.

(+) Até o facto de a metodologia orientada ao desenvolvimento de funcionalidades, prever o desenvolvimento de funcionalidades por uma única equipa de desenvolvimento - que vai distribuindo os seus membros por pequenos grupos de implementação -, reflecte o tipo de funcionamento implementável no CI-FCUL, devido ao reduzido número de elementos responsáveis pelo desenvolvimento.

(-) Uma das principais falhas do FDD no que diz respeito à sua adopção, relaciona-se com o facto de a metodologia não especificar um método de recolha de requisitos. Para além disso, os métodos que acaba por aconselhar para atingir esse fim (esquemas UML e pedaços de código aplicacional), são demasiado complexos para serem usados junto de um típico cliente de uma aplicação a desenvolver no CI-FCUL.

(+) No entanto, mesmo no que a este processo de recolha de requisitos diz respeito, é sugerido pelo método FDD, a formulação de casos de uso feita em conjuntopor membros da equipa de desenvolvimento e pelo cliente. Este processo vai de encontro ao processo utilizado pelo método XP (Programação

Extrema) com o uso das típicas "User Stories" (Histórias de Utilizador) para descrição de funcionalidades do sistema a implementar.

(-) Outro dos pontos menos atractivos desta metodologia é o facto de ter um grande número de papéis para atribuição aos seus diferentes membros. Mesmo contemplando a hipótese de cada membro desempenhar múltiplas funções, acaba por se tornar excessivo num pequeno grupo de desenvolvimento como o CI-FCUL, uma vez que a maior parte dos papéis serão sempre atribuídos a uma única pessoa, e uma vez que não contempla papéis simples como o de simples programador, exigindo que um programador, seja sempre, um proprietário de classe ("class owner") ou um perito de domínio ("Domain Expert"). A simplicidade ao nível dos papéis será também essencial a uma facilitada adopção de uma metodologia de desenvolvimento no Centro de Informática.

Conclusão

Após todos estes factores positivos e negativos terem sido identificados, pode-se dizer que a metodologia de desenvolvimento FDD, representa uma ideia bastante próxima da que se quer implementar no CI-FCUL, mas que no entanto, devido à sua demasiada complexidade, e falta de especificidade no processo de arranque de qualquer projecto, não é, no seu todo, ideal para adopção neste grupo.

De qualquer forma, a metodologia adoptada deverá sempre conter ideias resultantes desta metodologia e inclusivamente inspirar-se nas suas diferentes fases do processo de desenvolvimento, bem como nos seus diferentes papéis e responsabilidades.

Processo Racional Unificado (RUP)

A forte ligação a casos de uso para modelação de requisitos e para construção de sistemas, foi também por si só, ponto de partida para um tipo de processo ágil de desenvolvimento denominado Processo Racional Unificado (RUP). RUP representa uma aproximação iterativa de desenvolvimento para sistemas orientados a objectos, com uma ligação forte ao UML. É por esta razão, um tipo de processo muito mais simples que todos os outros que tenho vindo a resumir neste documento, e particularmente útil na adaptação conjunta com outros tipos de metodologias de desenvolvimento ágil.

Processos RUP:

O ciclo de vida do processo racional unificado, divide-se em 4 fases típicas, e cada uma destas por sua vez se divide num conjunto de iterações que se podem estender durante o período de 2 semanas a extensos 6 meses (Figura 54) ***[Kruchten, 2000]*:**

Incepção: Fase em que todos os objectivos e necessidades do projecto são inteiramente definidas, sendo identificados os casos de uso críticos que permitirão guiar o desenvolvimento de funcionalidades para o projecto.

Elaboração: Toda a base da arquitectura do software é elaborada com base na análise de todo o domínio do problema. O plano do projecto é então claramente definido e a infra-estrutura e o ambiente de desenvolvimento são descritos ao pormenor. Culmina com a criação de um protótipo executável da aplicação que reflecte todos os pormenores analisados. Nesta fase, determinam-se ainda os riscos principais no desenvolvimento do projecto e é comum haver grande recorrência a ferramentas automáticas como forma de poupar nos recursos e de simplificar o desenvolvimento.

Construção: Todas as funcionalidades da aplicação, definidas anteriormente, são desenvolvidas e integradas, após os testes devidos, no protótipo inicialmente construído. Este protótipo vai sendo "enriquecido" de funcionalidades, iteração após iteração. Existe grande ênfase no controlo de custos, de tempo (calendários) e de qualidade.

Transição: Fase atingida apenas quando o projecto é considerado consistente e rico o suficiente para ser lançado. Toda a fase de transição do projecto é baseada em respostas dadas pelos utilizadores às diferentes funcionalidades desenvolvidas, na correcção dos problemas encontrados e no desenvolvimento de funcionalidades pendentes. No final toda a documentação tendo em vista o utilizador é produzida.

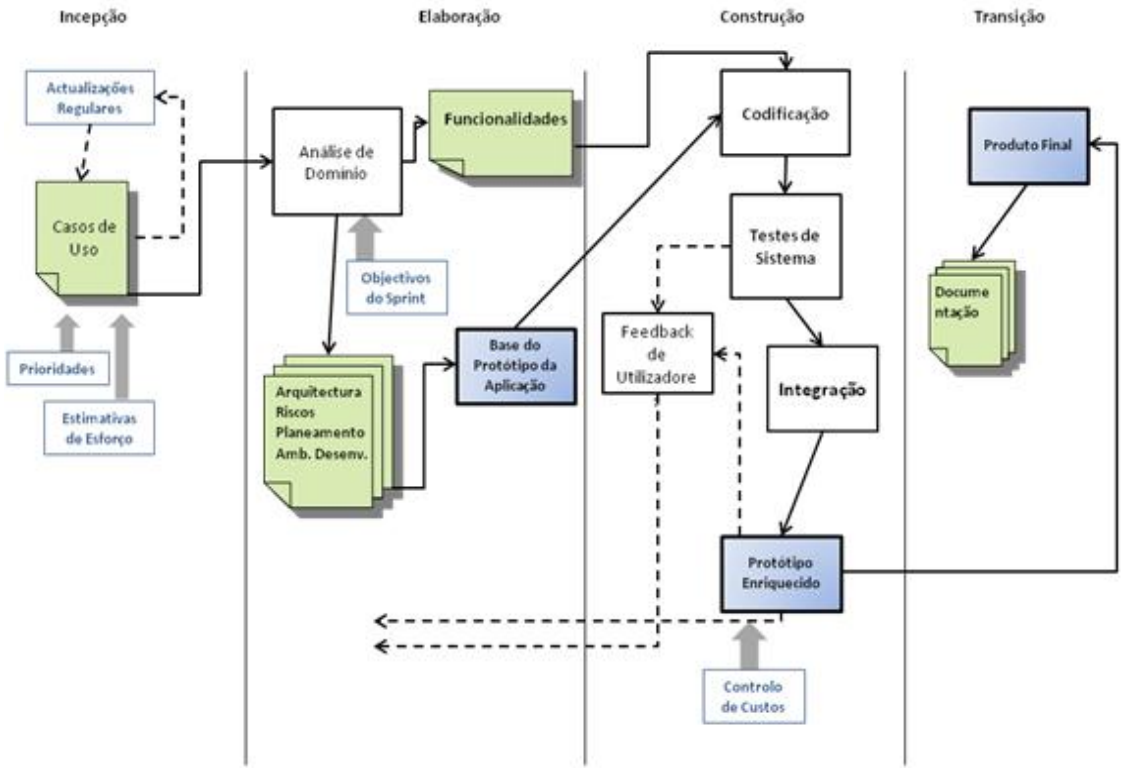


Figura 54: Esquema do Processo de Desenvolvimento de RUP

Pode-se ainda considerar que durante todas estas fases existe um conjunto de tarefas que são sempre cumpridas, são elas: Modelação do negócio; Levantamento e análise de requisitos; Implementação; Período de testes; Configuração e gestão de mudanças; Gestão do projecto e Gestão do ambiente. ***[Kruchten, 2000]*:**

De todas estas tarefas há a destacar duas menos familiares relativamente às restantes metodologias já descritas, assim:

Modelação do negócio: Tem por objectivo, assegurar que as necessidades do cliente são verdadeiramente atendidas, e que o produto final do desenvolvimento reflecte as suas ideias iniciais.

Gestão do ambiente: Destina-se a oferecer suporte ao trabalho em desenvolvimento, tendo assim por objectivos a implementação e configuração da metodologia adoptada, a selecção, desenvolvimento e aquisição de ferramentas e de hardware, bem como o treino de toda a equipa nos diversos recursos adoptados.

Papéis e Responsabilidades RUP:

De forma a simplificar todo o desenvolvimento, no processo racional unificado existe um papel para cada actividade no processo de desenvolvimento. Definem-se por isso 13 papéis denominados "trabalhadores", dos quais se destacam ***[Kruchten, 2000]*:**

Analista de Processos de Negócio: Tem a função de liderar e coordenar o processo de definição dos casos de uso do negócio posteriormente distribuídos pelos diversos ***Desenhadores de Negócio*.** Define o modelo de objectos de todo o negócio e descreve processos e actores responsáveis.

Revisor de Modelo de Negócio: Como o próprio nome indica, individualidade com a tarefa de rever todos os artefactos produzidos pelos analistas e desenhadores de negócio, bem como de identificar falhas nos mesmos.

"Toolsmith" (Artifice): Programador que desenvolve ferramentas para suporte a todo o desenvolvimento, servindo estas tanto para testes de funcionalidades, como para esboços de desenhos ou arquitecturas.

Para além destes, os típicos papéis de **Gestor de Projecto, Programador, "Tester", Cliente** estão logicamente presentes na metodologia.

Uma vez que toda a metodologia RUP é definida de forma muito subjectiva dependendo do projecto em desenvolvimento, o número de pessoas necessárias para assumirem as responsabilidades ou papéis necessários ao projecto pode variar de forma considerável.

Práticas, Adopções e Experiências RUP:

Por trás de toda a estrutura desta metodologia existem seis práticas que a caracterizam como metodologia ágil de desenvolvimento, são elas:

- **Desenvolvimento Iterativo:** Tal como todos os processos característicos de metodologias ágeis de desenvolvimento, o produto é desenvolvido em pequenos incrementos e curtas iterações, permitindo identificação prematura de riscos e problemas, e respectiva resolução.

- **Gestão de Requisitos:** A identificação, prioritização e filtragem de requisitos que mudam com o tempo ou com o ambiente, é uma tarefa indispensável a uma metodologia ágil. A definição adequada dos requisitos do projecto permite melhorar a comunicação em todo o processo de desenvolvimento.

- **Uso de Arquitecturas de Componentes:** A implementação de uma arquitectura baseada em componentes garante maior flexibilidade a todo o processo. O isolamento das componentes mais susceptíveis de mudança garante uma maior segurança, e a reutilização de componentes permite grande poupança de tempo de desenvolvimento de funcionalidades já conhecidas. Nesta área, com o uso de ferramentas automáticas de implementação (como são o caso dos CMS - gestores de conteúdo), a reutilização e implementação de componentes é facilitada em grande escala.

- **Software de Modelação Visual:** A prática mais característica de toda a metodologia, é a que se relaciona directamente com a construção de modelos, fazendo uso da linguagem UML (modelação visual). Ferramentas UML, permitem uma captura universal do sistema e através da sua descrição visual e consequente exposição do domínio do projecto, facilitam a comunicação, com e entre, todas as partes envolvidas no processo.

- **Verificação de Qualidade:** Uma preocupação constante com a qualidade do produto em desenvolvimento, traduz-se em grande número de testes durante todo o processo iterativo de desenvolvimento, tornando mais rápida a identificação de defeitos ou falhas no produto.

Como se acabou de ver, esta não é uma metodologia que obrigue a grandes exigências em termos de práticas, recursos, ou formas de implementação, baseia-se apenas na ideia do uso de ferramentas UML como forma de modelar o processo inicial de todo um levantamento de requisitos e definição do domínio de um projecto, em formas de controlo de variáveis em constante mudança e no uso de ferramentas de implementação automática, possibilitando ainda alguma agilidade dentro destas práticas. Por esta mesma razão, RUP poderá não ser considerado totalmente uma metodologia ágil, mas pode ser adoptado frequentemente por todo o tipo de metodologias de desenvolvimento deste tipo, podendo ser adoptado tanto na sua totalidade como apenas em parte, aproveitando algumas das suas práticas mais características.

Assim, e no capítulo da adopção e experimentação do método, convém ainda referir que:

- O Processo Racional Unificado foi até ao momento implementado e adoptado em muitas organizações, devido ao facto de ferramentas populares existirem no que diz respeito a software relacional (UML) e de desenvolvimento simples (CMS) tendo estes tipos de ferramentas de desenvolvimento um custo relativamente baixo.

- O ponto negativo de RUP (mas também principal razão para a sua utilidade), tal como já foi referido, é o facto de falhar no fornecimento de guias claros de implementação, deixando todo esse processo ao utilizador e às respectivas ferramentas e formas de implementação nas mesmas, concentrando-se na modelação de processos de negócio, na definição de requisitos e nas fases de análise e/ou desenho das aplicações ***[Ambler, 2002b].***

Probabilidades de Aceitação no CI:

A melhor forma de perceber as hipóteses de adopção ou aceitação desta metodologia de desenvolvimento de aplicações no CI-FCUL é enumerar os seus pontos positivos e negativos para o mesmo:

(+) O típico desenvolvimento iterativo característico deste tipo de metodologias reflecte-se também de uma forma muito forte no RUP através da construção inicial de um protótipo base que vai sendo enriquecido com módulos e funcionalidades implementadas. Isto permite ter sempre uma ideia do aspecto geral da aplicação e ir verificando o cumprimento dos objectivos dos requisitos, passo a passo.

(+) O grande controlo de custos, tempo e qualidade que é evidenciado nesta metodologia torna-a bastante atractiva para órgãos de desenvolvimento em que urge o controlo de despesas durante o projecto e em que é essencial respeitar os prazos de cada projecto em desenvolvimento.

(+) A preocupação que existe na total documentação do projecto assim que o seu desenvolvimento é concluído, permite o aproveitamento futuro de algumas técnicas utilizadas ou o proveito de problemas resolvidos no decorrer do mesmo, tal como permite a futuros desenvolvedores que não tiveram contacto original com o projecto, puderem ser responsáveis por desenvolvimentos parecidos, ou extensões sobre este.

(+) Também o treino constante efectuado por toda a equipa de desenvolvimento como parte do próprio processo de desenvolvimento de cada aplicação, representa um aspecto muito importante e vantajoso nesta metodologia, permitindo que a equipa de desenvolvimento se mantenha actualizada sobre novas técnicas de implementação.

(+) O facto de toda a revisão de funcionalidades englobada no processo de desenvolvimento de uma aplicação, ser levada a cabo pelos próprios clientes da aplicação, garante a satisfação dos objectivos do projecto, tal como serve para manter os clientes actualizados e satisfeitos durante o desenvolvimento da aplicação.

(-) O principal factor de impedimento para a adopção desta metodologia como metodologia de desenvolvimento do CI-FCUL, é o facto de esta ser principalmente útil em projectos ou equipas de desenvolvimento que trabalham essencialmente com ferramentas de implementação automática de funcionalidades (como gestores de conteúdos Web com módulos de desenvolvimento). É por isso mais apropriado a pequenos projectos com funcionalidades típicas e para equipas de desenvolvimento que usam ferramentas deste género para total implementação de serviços. Não é por esta razão, tipicamente aplicável a projectos mais extensos e com funcionalidades mais complexas, não servindo assim inteiramente o Centro de Informática da FCUL.

(-) Outro factor negativo, prende-se com o uso quase exclusivo de UML para recolha e análise de requisitos, dificultando o envolvimento do cliente no processo ou a relação entre este e a equipa de desenvolvimento. Existem formas mais simplificadas e acessíveis para o cliente, de chegar a requisitos e à ideia geral de um sistema que deve, na maior parte das vezes, ser descrito pelo próprio cliente.

Conclusão

Embora conte com muitos pontos positivos, esta metodologia não é considerada eficaz para adopção no CI-FCUL uma vez que não é adequada para o tipo de projectos e serviços que se pretende desenvolver, não sendo possível desenvolver todas as aplicações tendo como base uma ferramenta de desenvolvimento sistemático (tipo CMS), havendo grande número de serviços com funcionalidades únicas que requerem uma programação de raiz, e por isso requerem outro tipo de ferramentas.

Método de Desenvolvimento de Sistemas Dinâmicos (DSDM)

O Método de Desenvolvimento de Sistemas Dinâmicos, denominado frequentemente como DSDM, é um processo que segue princípios de interacção activa entre utilizadores, fornecendo às equipas de desenvolvimento poder suficiente para tomarem decisões, realizando contínuos testes distribuídos por todos os períodos de desenvolvimento e apoiando-se na construção frequente de protótipos e versões do produto em desenvolvimento. DSDM constitui-se como uma das plataformas mais usadas para rápido desenvolvimento de aplicações (RAD), bem como também é vastamente utilizada no controlo de métodos de desenvolvimento deste tipo.

Ao contrário da grande maioria dos métodos ágeis de desenvolvimento, é seu principal objectivo corrigir tempo e recursos gastos no processo de desenvolvimento, ajustando a quantidade de funcionalidades de acordo com os recursos disponíveis. Não se preocupa assim em implementar todas as funcionalidades inicialmente, mas primeiramente em respeitar os limites de tempo e recursos, desenvolvendo as funcionalidades possíveis. ***[Stapleton, 1997]***

Processo DSDM:

Em termos de processo de desenvolvimento, DSDM divide-se em 4 fases (Figura 55) ***[Stapleton, 1997]***:

1. **Estudos de Viabilidade e de Negócio:** Percebe-se a forma como o projecto tem de ser levado a cabo e sua viabilidade com este método (possibilidades técnicas e de negócio, probabilidade de ser levado a bom porto e riscos corridos). Leva em conta o tipo de projecto, sua organização e tipo de pessoal envolvido. É gerado um relatório de viabilidade e definido um plano de desenvolvimento. É realizado apenas uma vez por projecto e demora poucas semanas, podendo envolver o cliente da aplicação.

2. **Modelo Funcional de Iteração**: Em cada iteração, são planeados os conteúdos e planos de acção para estes e são analisados os resultados para futuras iterações. Análise e codificação são levadas a cabo construindo protótipos que vão ganhando qualidade até poderem ser incluídos no sistema final. É produzido um Modelo Funcional do sistema, contendo o código do protótipo e os modelos de análise e são ainda produzidos: uma lista prioritizada de funções implementadas, documentos de revisão do protótipo (com comentários de utilizadores), requisitos não-funcionais, e análise de riscos futuros.

3. **Desenho e Montagem de Iteração**: O sistema é construído na sua maior parte, sendo testado e garantido que preenche o conjunto mínimo de requisitos acordados inicialmente. Todo o desenvolvimento é baseado nos comentários dos utilizadores e dos clientes da aplicação.

4. **Implementação:** O sistema é transferido do ambiente de desenvolvimento para o ambiente de produção. Os utilizadores da aplicação recebem treino para a sua utilização e são produzidos Manuais de Utilizadores e Relatório de Revisão do Projecto.

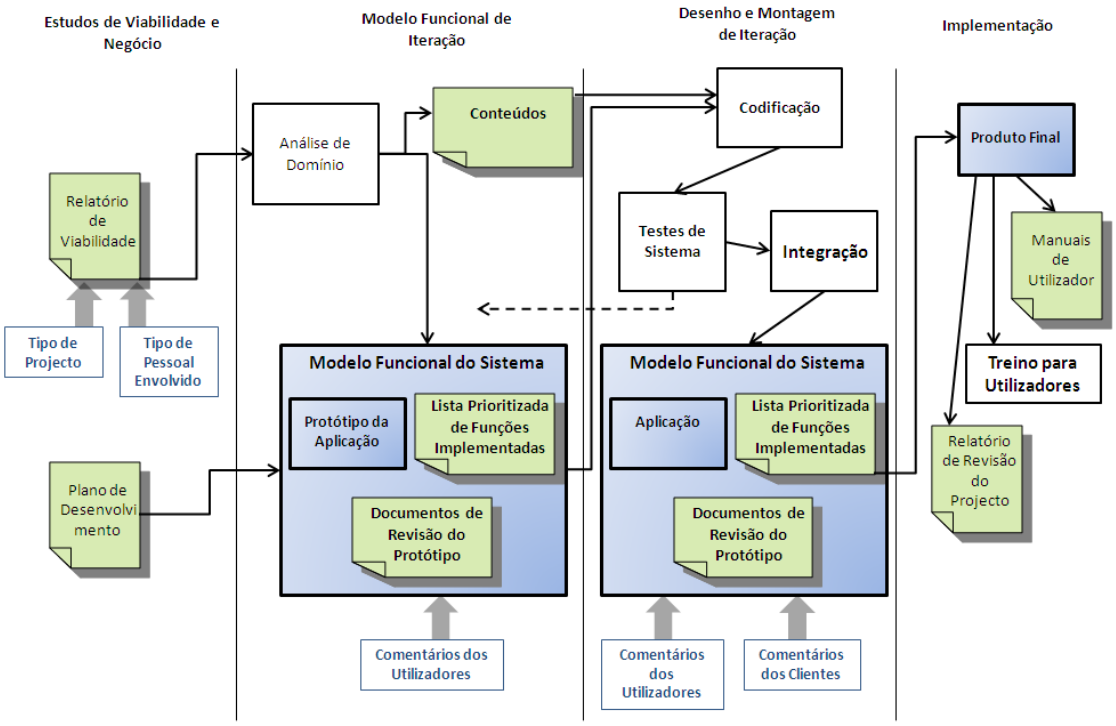


Figura 55: Esquema do Processo de Desenvolvimento de DSDM

Enquanto a primeira e últimas fases do processo só são realizadas uma vez, as restantes fases intermédias constituem o processo iterativo de desenvolvimento e enquanto houver funcionalidades para implementar, as duas fases funcionam como um ciclo.

A grande característica da metodologia DSDM, é a forma como para cada iteração (compreendida pelas 3 fases acima indicadas) é atribuído um "espaço de tempo" e sempre que este tempo se esgota, a iteração é considerada finalizada. Desta forma a duração típica de tempo para cada iteração é de poucos dias a poucas semanas.

Papéis e Responsabilidades DSDM:

Em termos de papéis e responsabilidades o DSDM define 16 papéis para utilizadores e desenvolvedores. Dessa imensidão de papéis, destacam-se os seguintes ***[Stapleton, 1997]***:

Desenvolvedor e Desenvolvedor Sénior: Compreende papéis conhecidos como Analista, Desenhador, Programador e "Tester" e desenvolvedores com mais experiência (seniores) têm normalmente papéis de um risco e dificuldade mais elevados.

Coordenador Técnico: Responsável por definir a arquitectura do sistema e por garantir a qualidade técnica do projecto (sendo por isso responsável pelo seu controlo geral).

Utilizador Embaixador: Tem a função de trazer a sabedoria da comunidade de utilizadores ao projecto e transmitir aos outros utilizadores o progresso do projecto.

Visionário: É um utilizador participante que detém a percepção mais apurada dos objectivos de negócio, do sistema e do projecto. Aquele que é normalmente responsável pela ideia inicial de construção do projecto.

Patrocinador: Utilizador com autoridade financeira e responsabilidade no projecto e que detém o poder principal de tomar as decisões.

Práticas, Adopção e Experiências:

Para que todas as fases do processo se possam traduzir na construção de aplicações consistentes, existem algumas práticas típicas de uma metodologia ágil de desenvolvimento que é necessário seguir: ***[Stapleton, 1997]***

- **Envolvimento activo de utilizadores:** Utilizadores têm de participar tanto no levantamento de requisitos para arranque do projecto, como na revisão das várias funcionalidades que vão sendo implementadas.

- **Equipas com poder de decisão:** Para além do patrocinador, também as equipas têm que deter alguma poder de decisão de forma a satisfazerem requisitos indispensáveis.

- **Desenvolvimento Incremental e iterativo**: 1 protótipo a cada curto ciclo de desenvolvimento é indispensável para a apresentação de resultados.

- **Mudanças reversíveis:** Tem que ser possível voltar atrás numa acção em qualquer fase do projecto**.**

- **Testes contínuos durante o desenvolvimento:** Testes permitem corrigir incoerências e problemas de desenvolvimento e ir equilibrando o funcionamento geral da aplicação.

- **Atitude colaborativa e cooperativa entre todos os envolvidos:** Essencial haver uma filosofia de trabalho de equipa. Ninguém desenvolve por sua conta e risco e tem que haver grande debate de ideias para aproveitamento ideal de recursos e tempo.

Embora não haja grandes relatos de uso desta metodologia, sabe-se que DSDM:

- É uma alternativa fiável para o rápido desenvolvimento de aplicações em meios muito limitados em termos de recursos financeiros/humanos e tempo (Sistemas de Negócio).

- Revela-se muito pouco eficaz no desenvolvimento de raiz de aplicações e funcionalidades, sendo preferível em situações em que a equipa de desenvolvimento já encontra blocos implementados ao seu dispor e necessita de adicionar funcionalidades ao serviço, ou desenvolver um serviço semelhante.

- É aplicada tanto em pequenos projectos como a projectos maiores em que o sistema é dividido em vários projectos mais pequenos e está dependente normalmente de uma equipa constituída

por duas (utilizador e desenvolvedor) a seis pessoas. Podem ainda existir várias equipas num mesmo projecto.

Probabilidades de Aceitação no CI:

É de certa forma, fácil de perceber a impossibilidade de uso de uma metodologia como a acabada de descrever no organismo como o CI-FCUL, no entanto, para confirmarmos essa ideia, é importante fazer um levantamento dos factores positivos e factores negativos do uso do DSDM:

(+) Antes de mais, é visível a forma como a partir desta metodologia é possível respeitar o tempo e recursos disponibilizados para um projecto (para além de ser documentável todo o processo de desenvolvimento). Através do patrocinador que detém poder último sobre qualquer matéria (porque é quem fornece precisamente os recursos) e de equipas responsabilizadas o principal objectivo da metodologia é não ultrapassar custos estipulados preocupando-se em segundo lugar por tentar implementar as funcionalidades mais importantes nesse espaço de tempo e de acordo com esses recursos.

(-) Contrapondo este ponto positivo, como no CI-FCUL é importante antes de mais que um projecto satisfaça inteiramente as funcionalidades identificadas na sua fase de recolha de requisitos, e satisfaça as aspirações do cliente, seria impossível a adopção de uma metodologia deste género. O facto de os recursos serem normalmente bastante limitados (equipa de desenvolvimento pequena e diversos projectos em simultâneo, só para avançar dois exemplos) provocaria que os serviços desenvolvidos ficassem em grande medida incompletos ou desprovidos das funcionalidades exigidas.

(-) Impõe-se referir, que o facto do próprio suporte à metodologia ser feito por uma empresa privada mediante pagamento de serviços, é impeditivo da adopção da mesma.

(-) Da análise dos vários papéis e responsabilidades típicos desta metodologia, conclui-se que o Patrocinador do projecto é a pessoa mais importante em todo o desenvolvimento, por ser quem dita o destino do mesmo. A inexistência de um Gestor de Projecto e a substituição deste por um Coordenador Técnico que apenas se preocupa em manter a qualidade técnica sem grande capacidade de decisão, torna esta metodologia impossível de adopção no CI-FCUL. Nem sempre a pessoa com autoridade financeira e gestora de recursos é a mesma que detém o conhecimento necessário para levar uma implementação de serviço a bom termo.

Conclusão

Assim, podemos confirmar que está posta de parte a adopção desta metodologia por parte do CI-FCUL por esta quantidade bastante grande de argumentos negativos.

Desenvolvimento de Software Adaptativo (ASD)

A metodologia de Desenvolvimento de Software Adaptativo (ASD) segue os princípios próprios dos métodos de desenvolvimento iterativos e ao mesmo tempo associa a estes a princípios de desenvolvimento rápido de aplicações**.** Foca-se dessa forma no de desenvolvimento complexo de sistemas vastos, encorajando um desenvolvimento incremental iterativo e com constante prototipagem. ***[Highsmith, 2000]***

ASD é explicitamente orientada a objectivos, ao invés de ser orientada a tarefas, focando-se acima de tudo na obtenção de resultados definidos como metas no início de cada projecto. Esta metodologia tem como principal objectivo fornecer uma plataforma com suficientes guias de desenvolvimento de forma a prevenir os projectos de caírem "em desgraça" em ambientes de desenvolvimento em constante mudança.

Processos ASD:

A metodologia de Desenvolvimento de Software Adaptativo é formada por um processo de desenvolvimento constituído por um ciclo trifásico (Figura 56) ***[Highsmith, 2000]***:

Especulação: Usado em vez de Plano, por sugerir insegurança. Divide-se em:

Iniciação do Projecto: Definição da missão do projecto em relação à qual todo o desenvolvimento será orientado e percepção da informação e dos requisitos necessários para executar o projecto.

Ciclo de Planeamento Adaptativo: Formação de um calendário geral do projecto, bem como de todos os objectivos do mesmo.

Colaboração: Evidencia a importância de uma equipa. ASD é explicitamente orientada a componentes, e nesta fase várias componentes estão sob desenvolvimento concorrente.

Aprendizagem: Necessidade de perceber e reagir a erros e mudanças.

Revisão de Qualidade: Componentes estão em constante refinamento para reflectirem informação em mudança. Nesta fase, fazem-se repetidas revisões de características do projecto de forma a permitir esse refinamento constante. É importante a presença do cliente e de um grupo de especialistas para discussões sobre futuras funcionalidades e sobre o progresso do desenvolvimento.

Lançamento: Importância de aprender lições para o futuro. Postmortems do projecto são importantes em projectos desenvolvidos sobre constantes mudanças.

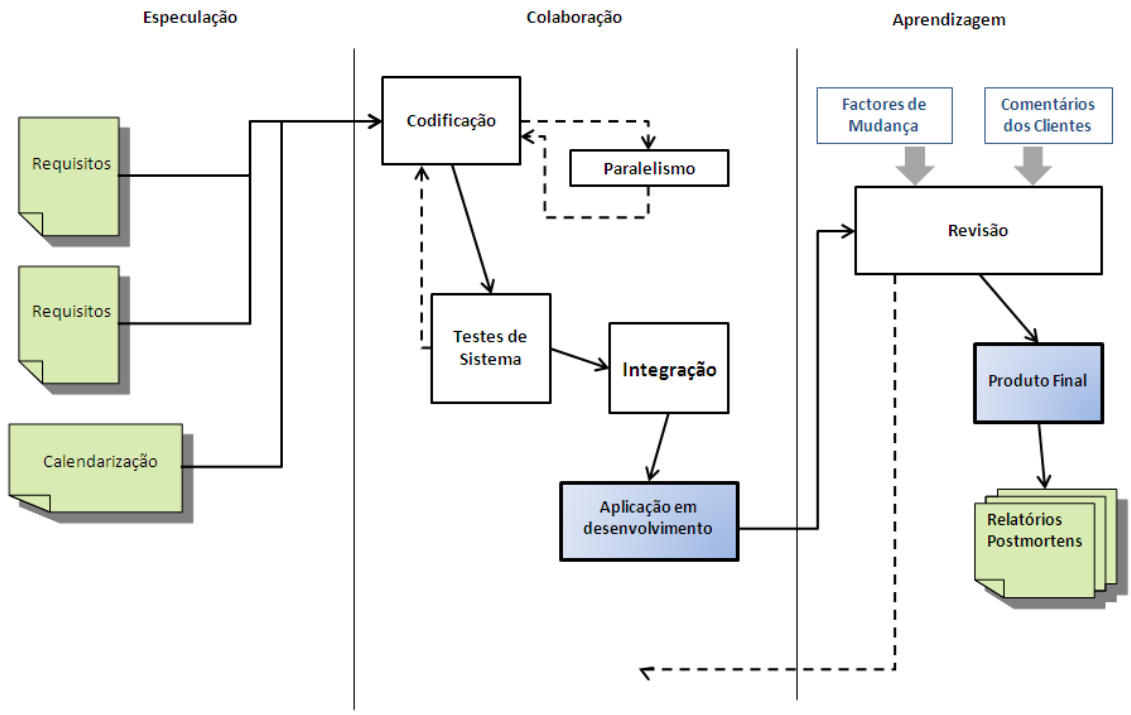


Figura 56: Esquema do Processo de Desenvolvimento de ASD

Papéis e Responsabilidades ASD:

Em termos de papéis e responsabilidades, a ASD dá especial ênfase à importância de equipas colaborantes e apoia-se por isso num grande trabalho de equipa. Embora não defina estruturas de equipas em detalhe, define cinco papéis característicos: ***[Highsmith, 2000]***

Patrocinador Executivo: Pessoa com responsabilidade geral sobre todo o projecto, e que é responsável pelo financiamento ou suporta os recursos utilizados.

Participantes Facilitadores: Pessoas responsáveis pelo planeamento e por liderar as sessões de desenvolvimento do projecto.

Gestor de Projecto: Participante Facilitador com maior poder de decisão e que detém a visão geral de todo o desenvolvimento.

Representante de Desenvolvimento: Responsável por reportar o progresso do projecto e explicar as funcionalidades desenvolvidas.

Representante de Cliente: Pessoa responsável pela interacção com o cliente da aplicação e que tem a tarefa de assegurar que as intenções deste figuram de facto no projecto, e que as funcionalidades em desenvolvimento o satisfazem.

Para além destes papéis característicos, existem papéis típicos como o de desenvolvedor que se pode dividir em Programador, Tester, Consultor, etc.

Práticas, Adopção e Experiências:

Na metodologia de Desenvolvimento de Software Adaptativo destacam-se práticas comuns à maioria dos métodos ágeis de desenvolvimento, bem como outras práticas mais características do próprio método:

- **Desenvolvimento Iterativo:** O método cascata só funciona bem em ambientes bem entendidos e bem definidos, por essa razão neste método existe especial foco em "refazer funcionalidades" em vez de as "fazer bem à primeira". O desenvolvimento por fases é por isso essencial para ir refinando funcionalidades e ir refinando aplicações, contemplando factores de mudança.

- **Tolerância a Mudanças:** A constante avaliação a componentes desenvolvidas que podem sofrer alterações é essencial em todo o processo de tolerância a mudanças. Sempre que alguma funcionalidade é modificada como resultado do ambiente, deve-se garantir a estabilidade da aplicação no seu funcionamento geral.

- **Orientação ao Risco:** Itens ou componentes de alto risco devem ser desenvolvidas o mais cedo possível no processo de desenvolvimento, de forma a minimizar problemas mais tarde.

- **Limites de Tempo:** A ambiguidade no desenvolvimento pode ser evitada através da fixação regular de prazos de implementação para cada componente, ou para componentes especiais. Isto obriga a que sejam feitos alguns compromissos desde cedo no projecto, evitando que este acabe por "andar à deriva".

- **Orientação a Objectivos:** Não interessa como a funcionalidade é implementada, desde que os objectivos da aplicação sejam cumpridos e desde que o software desenvolvido seja funcional.

- **Revisões de Grupo Baseadas no Cliente:** Para garantir que mesmo com mudanças constantes no ambiente de desenvolvimento a aplicação reflecte as intenções do seu cliente, é importante que revisões de funcionalidades sejam efectuadas por membros da equipa em constante colaboração com o cliente.

No que diz respeito, a relatos de experiência e adopção desta metodologia convém referir:

- Os princípios e ideias por trás da metodologia ASD são razoáveis, mas poucas linhas de desenvolvimento são fornecidas de facto para por o método em prática. Na realidade este método exige que outros métodos sejam usados/adoptados simultaneamente. As práticas da metodologia para além de serem difíceis de identificar, deixam muitos detalhes em aberto quanto à sua aplicação, sendo esta indefinição o problema mais significativo do método.

- ASD no entanto, não obriga ao uso de equipas localizadas no mesmo espaço para o desenvolvimento de aplicações, sendo que as dificuldades ao nível da equipa de desenvolvimento estão relacionadas essencialmente com as capacidades sociais, culturais e de trabalho em equipa dos seus diferentes membros. Como forma de atenuar estas dificuldades a metodologia adopta estratégias de partilha de informação, uso de ferramentas de comunicação e formas de introduzir gradualmente rigor no trabalho de um projecto.

Probabilidades de Aceitação no CI:

A percepção das possibilidades de adopção desta metodologia no Centro de Informática da FCUL é evidente, com a identificação dos seguintes aspectos positivos e negativos:

(+) O desenvolvimento de aplicações tendo em conta os objectivos das mesmas, e não a orientação a funcionalidades, tem um aspecto positivo tendo em conta o desenvolvimento no CI, que é, o de garantir que os objectivos do Cliente são de facto aquilo que se atinge no fim do desenvolvimento. Para o cliente, não interessa quantas funcionalidades ou como as funcionalidades são processadas, desde que o serviço seja capaz de fazer aquilo que ele exigiu desde o início.

(-) No entanto, e tendo em conta que na maior parte dos casos, os clientes não sabem bem aquilo que querem, guiar o desenvolvimento de aplicações no CI apenas pelos objectivos dos vários clientes, seria algo muito arriscado e que na maior parte das vezes levaria a uma não realização total dos serviços em desenvolvimento. Seria provável que um processo guiado desta forma, resultasse também numa aplicação completamente diferente do que na realidade seriam as intenções reais do cliente.

(-) Para além disso, no CI-FCUL quer-se ter um ambiente e uma equipa de desenvolvimento constante, e pretende-se que o desenvolvimento de aplicações se torne cada vez mais um processo mecânico. Não haverá pois, diferenças enormes entre os vários serviços desenvolvidos (ou pelo menos não se quer que haja) ou grandes mudanças nas especificações e no ambiente de desenvolvimento dos projectos, sendo por isso necessária uma metodologia de desenvolvimento baseado em processos bem definidos, baseado numa equipa bem definida e em ferramentas bem definidas, para o desenvolvimento de aplicações relativamente definidas.

(-) É também evidente, que a falta de linhas de desenvolvimento fornecidas por esta metodologia, no que diz respeito à forma de lidar com a mudança, à forma de levantamento de requisitos para arranque do projecto, ou mesmo à forma de constituição da equipa de desenvolvimento, representa uma falha clara e impeditiva da adopção do ASD no CI-FCUL.

Conclusão

Por todas as razões negativas adiantadas, a não adopção desta metodologia no Centro de Informática da FCUL é natural e justificada.

Apêndice B

Documentos Gerados na Metodologia

CIFCUL

Centro de Informática
Faculdade de Ciências
Universidade de Lisboa

Nome do Projecto
Plano de Projecto

Versão:
Designação:
Autor:
Ano:

Criação:
Revisões:

Plano do Projecto

Identificação de Projecto

Nome:	NOME DO PROJECTO				
Enquadramento:	CONTEXTO DO PROJECTO	Sigla:	INICIAISDOPROJECTO	Ano:	ANO DO PROJECTO

Plano de Acção:

Prioritização de Desenvolvimento

Ordem de Desen.	User Story Id	Descrição da Funcionalidade	Prioridade Cliente	Esforço Desen.	Prioridade Eq. Desenvol.	Versão de Inclusão
1	USID	DESCRICAO	1 a 5	X horas	1 a 5	1ª
2	…	…	…	…	…	…

Calendarização de Tarefas

Ordem de Desen.	User Story Id	Descrição da Funcionalidade	Data Começo Estimada	Data Fim Estimada	Tempo Desenvolvimento	Versão de Inclusão
1	USID	DESCRICAO	DATA	DATA	X horas	1ª
2	…	…	…	…	…	…

Atribuição de Recursos a Tarefas

Ordem de Desen.	User Story Id	Tarefa	Desenvolvedores (Membros Eq.Des.)	Ferramentas Utilizadas	Tempo Desenvolvimento
1	USID	DRESCRICAO	Nome (Cargo), Nome(Cargo) …	FERRAMENTA	X horas
2	…	…	…	…	…

Planeamento de Versões

Nº Versão	Data Início Estimada	Data Fim Estimada	Tempo Desenv.	Nº Func. Implementadas
1ª	DATA	DATA	Horas/Dias/Semanas	X (User Story Ids…)
…	…	…	…	…

Planeamento de Testes:

Funcionalidade em Teste			Casos de Teste			
Ordem Des. Original	User Story Id	Descrição	Descrição do Teste	Data Planeada	Cliente	Membros Eq. Desenvolvimento
1	USID	DESCRICAO	DESCRICAO	DATA	NOME (CARGO)	NOME (CARGO)…
…	…	…	…	…	…	…

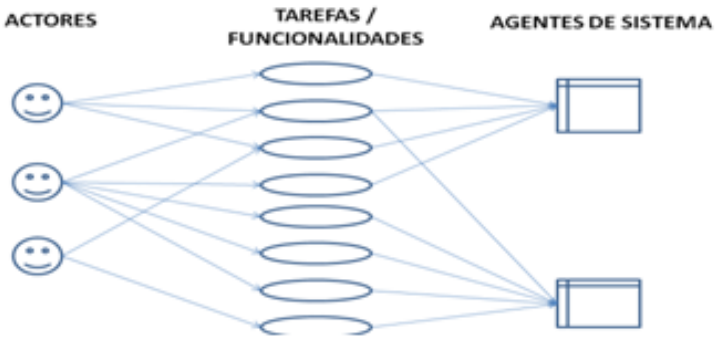
Planeamento de Revisões

Tipo de Revisão	Data da Revisão	Membros Envolvidos	Objectivo
REUNIÃO/DEBATE/CONJUNTO DE TESTES/ETC	DATA	NOME(CARGO), …	OJECTIVO…

Descrição Geral de Tarefas

Caso de Uso Geral

DESENHO DE CASO DE USO GERAL QUE FAÇA RELAÇÃO ENTRE ACTORES, TAREFAS e SISTEMA
TIPO:



Casos de Uso de Funcionalidades

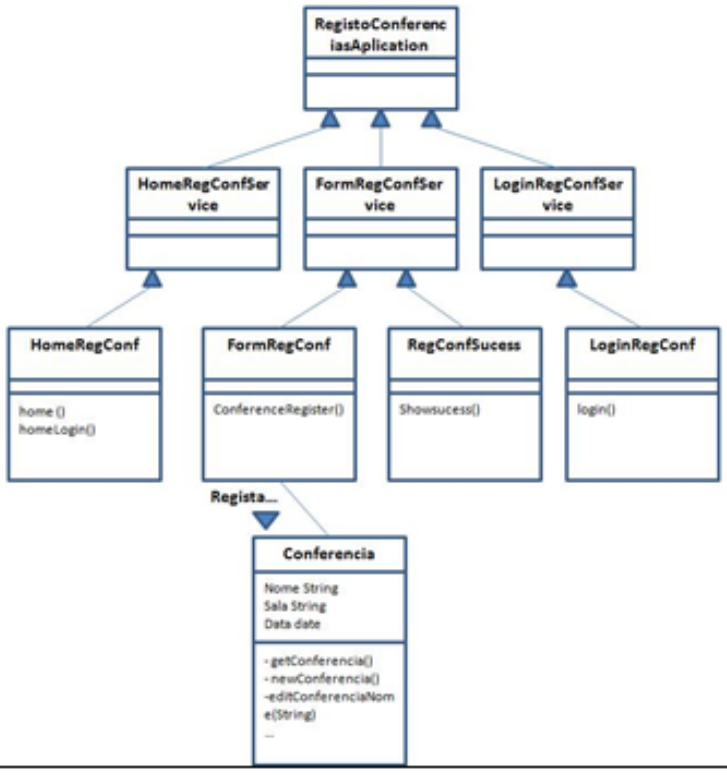
	ENUMERAÇÃO: NOME DO CASO DO USO	
Descrição do Caso de Uso:		
Sumário:	"Utilizador faz... para ... "	
Actores:	QUEM INTERVEM/REALIZA A ACÇÃO	
Pré-Condições:	CONDICOES QUE TEM DE SER SATISFEITAS ANTES DO INICIO DO CASO DE USO	
Pós-Condições:	CONDICOES QUE TEM DE ESTAR SATISFEITAS NO FIM DO CASO DE USO	
Prioridade:	PRIORIDADE DA FUNCIONALIDADE	
Cenário:		
1)	ACÇÃO	
a. b.	OPCÕES NA ACÇÃO OPÇÕES NA ACÇÃO	
2)	ACÇÃO	
a. b.	OPCÕES NA ACÇÃO OPÇÕES NA ACÇÃO	
3)	ACÇÃO	
a. b.	OPCÕES NA ACÇÃO OPÇÕES NA ACÇÃO	
Diagrama de Sequência: (?)		
DIAGRAMA DE SEQUÊNCIA		
Notas:		
NOTAS ADICIONAIS		
Integração:		
OPERAÇÕES NECESSÁRIAS PARA INTEGRAÇÃO NA APLICAÇÃO DA FUNCIONALIDADE (DEPENDÊNCIAS...ETC)		

	ENUMERAÇÃO: NOME DO CASO DO USO	
Descrição do Caso de Uso:		
Sumário:	"Utilizador faz... para ... "	
Actores:	QUEM INTERVEM/REALIZA A ACÇÃO	
Pré-Condições:	CONDICOES QUE TEM DE SER SATISFEITAS ANTES DO INICIO DO CASO DE USO	
Pós-Condições:	CONDICOES QUE TEM DE ESTAR SATISFEITAS NO FIM DO CASO DE USO	
Prioridade:	PRIORIDADE DA FUNCIONALIDADE	
Cenário:		
1)	ACÇÃO	
a. b.	OPÇÕES NA ACÇÃO OPÇÕES NA ACÇÃO	
2)	ACÇÃO	
a. b.	OPÇÕES NA ACÇÃO OPÇÕES NA ACÇÃO	
3)	ACÇÃO	
a. b.	OPÇÕES NA ACÇÃO OPÇÕES NA ACÇÃO	
Notas:		
NOTAS ADICIONAIS		
Integração:		
OPERAÇÕES NECESSÁRIAS PARA INTEGRAÇÃO NA APLICAÇÃO DA FUNCIONALIDADE (DEPENDÊNCIAS...ETC)		

Plano Técnico:

Diagramas de Classes

ESQUEMA QUE LISTE TODAS AS CLASSES A IMPLEMENTAR E SUA ESTRUTURA NA APLICAÇÃO
Exemplo:



Diagramas de Sequência
ESQUEMA QUE POSSIBILITE UMA PERCEPÇÃO DA SEQUÊNCIA DE TAREFAS A REALIZAR PARA CADA FUNCIONALIDADE

Estruturação de Funcionalidades - DIF2.0:

Funcionalidade		Objectos de Implementação				
User Story Id	Descrição da Funcionalidade	Nome da Aplicação	Nome do Serviço	Nome do Stage	Outros Objectos a Criar	Pormenores de Implementação
USID	DESCRICAO	QQCOISAAplication	QQCOISAService	QQCOISA	NOME - TIPO	OBSERVAÇÕES
RC14	Registar uma Conferência	RegistoConferenciasAplication	FormRegConfService	FormRegConf	Conferencia - Classe	- Conferência tem de ser definida ao nível da base de dados - Classe Conferência é gerada automáticamente pela DIF2.0 com reverse engeneering
RC15	Efectuar Login no Sítio de Marcação de Conferências	RegistoConferenciasAplication	LoginRegConfService	LoginRegConf	N/A	- Página de Sucesso de Login é a Home - Página de Falha de Login é a mesma com mostragem de erro.
RC01	Apresentar informação sobre registo de conferências	RegistoConferenciasAplication	HomeRegConfService	HomeRegConf	- Classe acessória responsável pela importação de conteúdos de outras páginas Web	
---	---	---	---	---	---	--

Estruturação de Funcionalidades - Drupal:

Funcionalidade		Módulos a Utilizar			Objectos de Implementação	
User Story Id	Descrição da Funcionalidade	Nome do Módulo	Função do Módulo	Pormenores de Implementação	Nome do Objecto	Tipo do Objecto
USID	DESCRICAO	NOME DO MODULO	FUNÇÃO DO MÓDULO	OBSERVAÇÕES	NOME	TIPO
RC16	Visualizar notícias de conferências marcadadas.	View, Panels	Views: Visualizar certo tipo de conteúdos filtrando pelas suas características. Panels: Colocar um conjunto de informações de tipos diferentes no mesmo bloco de conteúdo.	N/A	Vista_Noticias Painel_Noticias	View Panel
---	---	---	---	---	---	---

Gestão de Risco

Identificação de Riscos:				
Id Risco:	Risco:	Probabilidade de Ocorrência:	Impacto:	Fase de Ocorrência:
INICIAIS_DO_PROJECTO_RSCXXX	DESCRIÇÃO DO RISCO	Pouco Provável / Provável / Muito Provável	1/2/3/4/5	Todas/ (Pelo menos uma das 4)
--	--	--	--	--

Mitigação dos Riscos Identificados:	
Id Risco:	IDRISCO
Risco:	DESCRIÇÃO DO RISCO
Mitigação:	FORMAS SABIDAS DE EVITAR E COMBATER O RISCO
Monitorização:	FORMAS DE MONITORIZAR A TAREFA DE MODO A PERCEBER A SUA EVOLUÇÃO
Gestão:	FORMAS DE RESPONDER AO RISCO ASSIM QUE ESTE OCORRE
Id Risco:	IDRISCO
Risco:	DESCRIÇÃO DO RISCO
Mitigação:	
Monitorização:	
Gestão:	
Id Risco:	IDRISCO
Risco:	DESCRIÇÃO DO RISCO
Mitigação:	
Monitorização:	
Gestão:	

CIFCUL

Centro de Informática
Faculdade de Ciências
Universidade de Lisboa

Nome do Projecto
Revisão Técnica do Projecto

Versão:	
Designação:	
Autor:	
Ano:	
Criação:	
Revisões:	

Revisão Técnica do Projecto

Identificação de Projecto

Nome:	NOME DO PROJECTO				
Enquadramento:	CONTEXTO DO PROJECTO	Sigla:	INICIAISDOPROJECTO	Ano:	ANO DO PROJECTO

Acompanhamento do Desenvolvimento:

Funcionalidades Desenvolvidas e Canceladas							
Ordem Des. Original	User Story Id	Descrição da Funcionalidade	Data Começo	Data Fim	Tempo Desenv.	Versão Api.	Estado Desenvolvimento
1	USID	DESCRICAO	DATA	DATA	X horas	1ª	Implementado/Cancelado
--	--	--	--	--	--	--	

Observações de Desenvolvimento			
Ordem Des. Original	User Story Id	Descrição da Funcionalidade	Razões de Cancelamento
12	USID	DESCRICAO	OBSERVAÇÕES
--	--	--	--

Novas Funcionalidades Para Desenvolvimento			
Versão Api. para Inclusão	User Story Id	Descrição da Funcionalidade	Razões de Inclusão na Aplicação
VERSÃO	USID	DESCRICAO	OBSERVAÇÕES
--	--	--	--

Acompanhamento de Casos de Teste:

Casos de Teste Realizados:								
Funcionalidade em Teste			Casos de Teste					
Ordem Des. Original	User Story Id	Descrição	Descrição do Teste	Data Do Teste	Cliente	Feedback do Cliente	Membros Eq. Desenvolvimento	Aprovação
1	USID	DESCRICAO	DESCRICAO	DATA	NOME (CARGO)	OBSERVAÇÕES	NOME (CARGO)...	Sim / Não

CIFCUL

Centro de Informática
Faculdade de Ciências
Universidade de Lisboa

Nome do Projecto
Proposta de Projecto

Versão:
Designação:
Autor:
Ano:

Criação:
Revisões:

Versão:	Versao do Documento	Designação:	Designação da Versão	Autor:	Autor do Documento	Ano:	Ano do Documento

Proposta de Projecto

Identificação de Projecto

Nome:	NOME DO PROJECTO				
Enquadramento:	CONTEXTO DO PROJECTO	**Sigla:**	INICIAISDOPROJECTO	**Ano:**	ANO DO PROJECTO

Informação Básica:

Gestor:	NOME DA PESSOA (CARGO DA PESSOA)
Cliente:	NOME DA PESSOA (CARGO DA PESSOA)
Data de Início Estimada:	DD/MM/AAAA
Data de Fim Estimada:	DD/MM/AAAA
Duração Estimada:	X Dias/Semanas/Meses

Descrição Geral:
DESCRIÇÃO

Objectivos do Projecto:
1. OBJECTIVO 2. OBJECTIVO 3. OBJECTIVO 4. …

Versão:	Versao do Documento	Designação:	Designação da Versão	Autor:	Autor do Documento	Ano:	Ano do Documento

Requisitos:

User Stories:

Id:	Título:	Fonte:
Descrição		
Comentários:		Prioridade:
		Esforço Estimado:

Id:	Título:	Fonte:
Descrição		
Comentários:		Prioridade:
		Esforço Estimado:

Id:	Título:	Fonte:
Descrição		
Comentários:		Prioridade:
		Esforço Estimado:

Id:	Título:	Fonte:
Descrição		
Comentários:		Prioridade:
		Esforço Estimado:

Versão:	Versao do Documento	Designação:	Designação da Versão	Autor:	Autor do Documento	Ano:	Ano do Documento

Equipa de Desenvolvimento:			
Cargo do Trabalhador:	Nome do Trabalhador:	Carga Horária Semanal:	Tempo no Projecto:
TIPO DE CARGO	NOME DO TRABALHADOR	X horas	X dias /sem/meses

Ferramentas Suplementares a Utilizar na Implementação:			
Nome da Ferramenta:	Tipo de Tarefa Realizada:	Frequência/Fases de Utilização:	Membro da Equipa Responsável:
FERRAMENTA	TAREFA	FASE/FREQUÊNCIA	NOME DO RESPONSAVEL

Outros Detalhes do Projecto:

Custos Suplementares do Projecto:			
Tipo de Recurso:	Custo do Recurso:	Frequência/Fases de Utilização:	Membro da Equipa Responsável:
FERRAMENTA	CUSTO	FASE/FREQUÊNCIA	NOME DO RESPONSAVEL

Benefícios do Projecto:

1. BENEFÍCIO
2. BENEFÍCIO
3. BENEFÍCIO
4. ...

Documentos Relacionados com o Projecto:		
Nome do Documento:	Fases do Projecto Referente:	Observações:
NOME DO DOCUMENTO	FASE/FREQUÊNCIA	TEXTO

CIFCUL

Centro de Informática
Faculdade de Ciências
Universidade de Lisboa

Nome do Projecto

Relatório de Lançamento de Projecto

Versão:	
Designação:	
Autor:	
Ano:	
Criação:	
Revisões:	

Versão:	Versao do Documento	Designação:	Designação da Versão	Autor:	Autor do Documento	Ano:	Ano do Documento

Relatório de Lançamento

Identificação de Projecto

Nome:	NOME DO PROJECTO				
Enquadramento:	CONTEXTO DO PROJECTO	Sigla:	INICIAISDOPROJECTO	Ano:	ANO DO PROJECTO

Guia de Utilização:

Utilização Básica da Aplicação

Exemplos de Utilização	
Descrição da Tarefa	Realização da Tarefa
DESCRICAO	IMAGENS EXEMPLIFICATIVAS DA TAREFA

Conclusões Técnicas:

Lista de Funcionalidades Implementadas			
Nome	Descrição e Funcionamento da Funcionalidade	Quem a Realiza	Data Inclusão
NOME	DESCRICAO	UTILIZADOR	DATA
--	--	--	--

Lista de Funcionalidades por Implementar (Canceladas)				
Ordem Desenv. Original	User Story Id Original	Descrição da Funcionalidade	Data Começo Prevista	Razão de Não Implementação
1	USID	DESCRICAO	DATA	OBSERVAÇÃO
--	--	--	--	

Considerações Finais Sobre Desenvolvimento

Versão:	Versao do Documento	Designação:	Designação da Versão	Autor:	Autor do Documento	Ano:	Ano do Documento

Considerações Finais Do Cliente

Possíveis Extensões Futuras ao Projecto

Apêndice C

Estudo de Sistemas de Gestão de Conteúdos (CMS) para o CI-FCUL

O estudo seguinte, está organizado de uma forma relativamente simples, sendo que para cada gestor de conteúdos abordado é fornecida uma descrição base do mesmo ("O que é?"), são enumeradas as suas características mais importantes para uso e adopção no Centro de Informática da FCUL ("Características principais para uso no CI"), e são avançadas as principais razões para adopção ou não adopção deste.

Alfresco

O que é?

Alfresco CMS é um gestor de conteúdos recente mas que tem tido grande impacto no mundo empresarial, tendo sido revelado na altura do seu lançamento como o primeiro ECM (Enterprise Content Manager – Gestor de conteúdos empresariais) open-source. No entanto o facto de existirem já outros CMS do mesmo género e o facto de nem todos os conteúdos deste CMS serem gratuitos necessitando do pagamento de uma licença, joga à partida contra a sua larga utilização comunitária.

Fundado em 2005 por John Newton e John Powell, este gestor empresarial de conteúdos destaca-se na força estrutural da sua gestão de documentos, e tem como ponto mais fraco o seu ténue suporte de conteúdos web e respectivo desenvolvimento (havendo inclusivamente relatos que os módulos referentes a este tipo de desenvolvimento terão sido descontinuados, tendo sido recomendada outra plataforma independente para obter os mesmos fins.)

Características principais para uso no CI

Criação e Edição de Conteúdo:

Segundo o sítio oficial do Alfresco CMS uma das suas principais características é a forma dinâmica e eficiente como é possível um utilizador criar o tipo de conteúdo que deseja a partir desta plataforma. No seu comportamento suposto, a criação de conteúdos é feita através do preenchimento dos diferentes campos que constituem cada tipo de informação criada e através do uso de uma ferramenta centrada no utilizador, que permite através de operações de arrastamento de componentes e selecção de campos, criar conteúdos de todo o género nos locais pretendidos das páginas Web criadas. No entanto, como já referi, esta ferramenta denominada "Alfresco Studio" (Figura 57) revelou-se inutilizável na experimentação deste gestor de conteúdos e por toda a comunidade existem relatos que dão como "perdida" esta funcionalidade.

Figura 57: Imagem de Alfresco Studio

No entanto, a criação de conteúdos dos mais variados tipos continua a ser possível através da interface de administração do Alfresco de forma semelhante à criação de ficheiros no Microsoft Windows.

Gestão:

Alfresco como sistema gestor de conteúdos é de uso simplificado devido as diversas formas de escrever e ler dados do sistema. Utilizando inclusive um protocolo CIFS (Microsoft's Common Internet File System) permite aos utilizadores de Windows (a maioria) construir um repositório de dados como uma normal rede de partilha de ficheiros, ficando o próprio sistema encarregue das operações de gestão, versionamento, e extracção de contextos, aliviando a carga de conhecimento sobre o simples utilizador.

Todo o controlo administrativo e apresentação de informação relativa aos dados no CMS são feitos através do seu cliente Web e da sua interface de administração (Figura 58). Neste, os utilizadores com as respectivas permissões podem definir regras ou fluxos de regras para os conteúdos, tal como podem gerir a segurança dos diferentes dados hospedados. No Alfresco existe o conceito de "Aspectos" como um conjunto de atributos ou capacidades que podem ser agregadas aos diferentes objectos de forma independente e portanto não se relacionando com aspectos de herança de propriedades. Esta é de facto a principal "força" deste CMS, uma vez que toda a criação e gestão de fluxos de tarefas, acessos e versionamento é simples, intuitiva e de um funcionamento tremendamente eficaz.

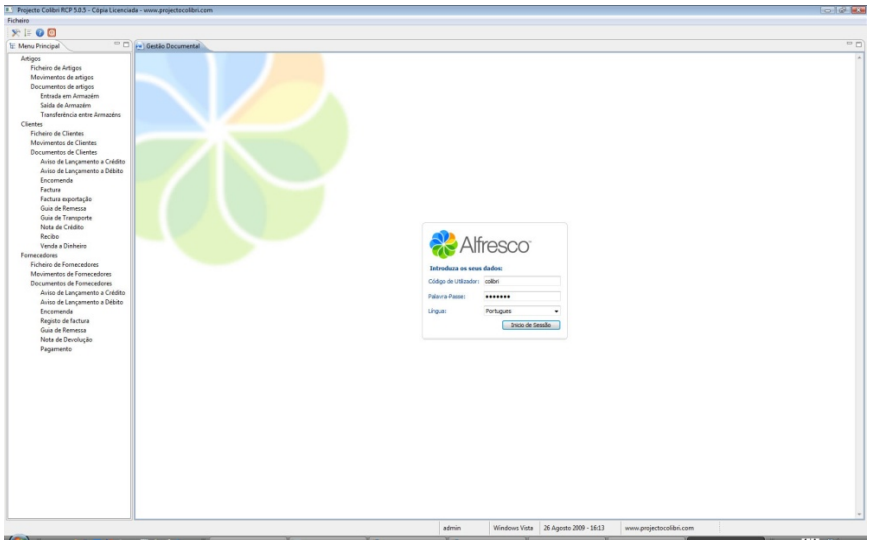


Figura 58: Foto da interface de administração do Alfresco CMS

Alfresco dispõe ainda de um módulo de partilha denominado "SHARE" que permite a gestão temporal e de recursos de um projecto, servindo como espaço colaborativo para todos os utilizadores e como forma de guiar um projecto em rede com a marcação de eventos, revisão de documentos, etc.

Apresentação:

No que diz respeito a formas de pesquisa dos diferentes conteúdos presentes no CMS existe uma compatibilidade com a ferramenta Open Office que permite extrair, transformar e apresentar informação presente em vários tipos de ficheiros de diferentes formatos.

No que diz respeito à apresentação propriamente dita, existe a definição de "pastas" e "espaços" de conteúdos que são apresentados ao nível da interface do cliente Web utilizando uma tecnologia de templates que usa linguagem FreeMarker para sua definição estrutural.

No entanto qualquer definição de templates de apresentação traduz-se num funcionamento instável da aplicação, tendo sido necessário recorrer a uma versão mais antiga da mesma para que esta funcionalidade fosse de facto devidamente apresentada, e mesmo nessa versão, o seu funcionamento traduziu-se em fraca estabilidade e muito propensa a erros.

Comunidade e Suporte:

Embora existam número relativamente grande de fóruns e sítios Web estilo Wikis que procuram dar suporte ao CMS, a comunidade Alfresco não é das mais ricas ao nível de documentação e suporte a erros da aplicação, sendo comum a identificação de problemas identificados por vários utilizadores, mas para os quais não existe uma resolução reportada ou suportada pela comunidade, ou pelos próprios fabricantes da aplicação. Isto deve-se em parte ao facto de a grande maioria dos módulos existentes para o sistema, serem desenvolvidos por parte dos próprios fabricantes e virem já de origem na própria aplicação.

Estes fabricantes estão no entanto abertos à inclusão de novos módulos de acrescento de funcionalidades sobre o sistema, desenvolvidos por membros da comunidade, bem como para a integração de novas soluções sobre problemas verificados com a sua utilização.

A reduzida quantidade de documentos de suporte ao uso e desenvolvimento de mecanismos e módulos sobre a aplicação são no entanto uma realidade, conduzindo à estagnação de novos módulos/recursos e ao "abandono" de diversas funcionalidades que originalmente fariam parte deste CMS (como o desenvolvimento de conteúdos Web, ou o desenvolvimento de templates de apresentação).

Problemas Identificados/Razões de Falha

O uso do CMS Alfresco no CI-FCUL não é uma opção válida devido principalmente às seguintes fraquezas:

- Reduzida orientação e suporte ao desenvolvimento de conteúdos Web, o que representa um dos principais objectivos da adopção do CMS.

- Comportamento instável da aplicação e dificuldades na instalação da mesma verificando-se um grande número de erros e procedimentos de estabilização sugeridos pela comunidade, com resultados pouco eficazes.

- Comunidade pouco activa, resolução de problemas precária e uso generalizado da aplicação apenas na componente de gestão de ficheiros, com abandono parcial no desenvolvimento de módulos relacionados com desenvolvimento de conteúdos.

OpenCMS

O que é?

OpenCMS é um sistema de gestão de conteúdos open-source escrito em linguagem Java, distribuído pela Alkacon e que requer o uso de Apache Tomcat. Baseia-se num ambiente baseado em browser que permite gestão e edição de conteúdos estáticos, gestão de utilizadores, gestão de fluxos de trabalho, gestão de versões, entre outras tarefas de menor importância.

OpenCMS foi lançado em 2000 e baseado no seu predecessor MhtCMS, e desde logo utilizado por organizações como o LGT Bank of Lichtenstein, BP South Africa e UNICEF Holanda.

Características principais para uso no CI

Criação e Edição de Conteúdo:

OpenCMS tem a sua grande força na forma como facilita o uso/edição de conteúdos de texto relativamente estático, a utilizadores não programadores, permitindo a actualização de conteúdo estrutural de um sítio de forma muito amigável, baseado em botões “edit me” (em português “edita-me”) que aquando seleccionados permitem a edição dos conteúdos em simples caixas de texto (Figura 59). Para além disto, esta plataforma permite criar diferentes tipos de conteúdos estáticos (variando no número de campos de texto, etc.) que são posteriormente editáveis a partir de operações muito simples de edição de texto. No entanto, a forma como são criados os diferentes conteúdos é mais complexa do que a forma como esses são editados. Isto é, para criação de um conteúdo para o site, é necessária a criação de um ficheiro, operações ao nível dos ficheiros e edição de linhas de programação que fazem com que este tipo de tarefa não esteja ao alcance do utilizador comum (embora esteja ao alcance de um utilizador com conhecimentos básicos de programação).

Figura 59: Edição de conteúdos no OpenCms

Assim, podemos concluir, que o OpenCMS é feito para edição de conteúdo e não propriamente para a sua criação por parte dos seus utilizadores comuns. Esta é uma das principais razões que levam a que este CMS não seja o ideal para adopção no CI-FCUL.

Gestão:

No que à gestão de conteúdos diz respeito, o OpenCMS apenas permite aos seus utilizadores a configuração de propriedades e elementos através da própria interface Web da aplicação, o que torna a gestão remota possível mas ao mesmo tempo dificulta de sobremaneira as tarefas de programação, uma vez que tudo tem que ser escrito sobre a forma de um bloco de notas, perdendo-se formas mais automáticas de implementar novos recursos em JSP.

No entanto esta será praticamente a única limitação desta plataforma de gestão de conteúdos, sendo possível através da interface de administração (Figura 60) realizar todo o tipo de tarefas de configuração, como é são os casos de configuração de espaços e pastas de ficheiros, gestão das bases de dados, gestão dos módulos instalados, configuração de ligações externas, calendarização de eventos de edição ou edição de conteúdos, acesso a sistema de procura de informação guardada no sistema e ainda configuração de perfis de utilizadores e seus respectivos direitos e deveres.

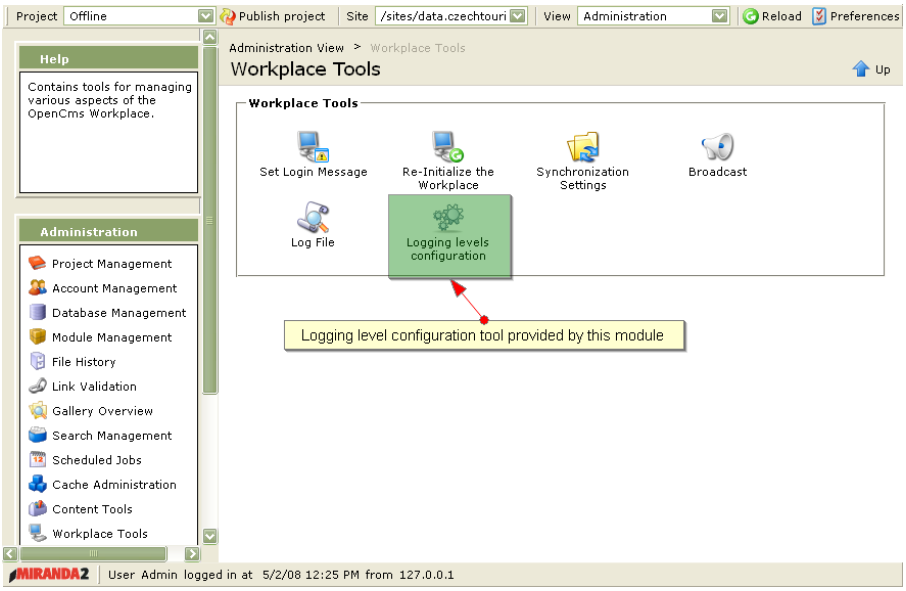


Figura 60: Interface de Administração do OpenCMS

Apresentação:

Este gestor de conteúdos, ao nível da apresentação, permite a adopção de sistemas de templates muito complexos – o que constitui uma vantagem, no entanto isto faz com que a própria criação e configuração dos templates de apresentação e de menus de navegação seja uma tarefa demasiado complexa - qualquer criação deste género exige bastante trabalho sobre código JSP - para o utilizador que se deseja como alvo.

Comunidade e Suporte:

Ao nível de documentação, este CMS apresenta alguns fóruns e comunidades que permitem simplificar processos de criação de conteúdos mais complexos, no entanto, com o passar do tempo, e com o surgimento de ferramentas que conferem maior liberdade ao utilizador comum e que estão ao mesmo orientados ao programador, estas comunidades foram reduzindo de dimensão e a aplicação foi estagnando nas novas funcionalidades de versão para versão. Esta realidade, deixa perceber que este CMS embora seja bastante poderoso é também demasiado complexa para ser uma ferramenta de futuro. E que existem melhores soluções no mercado.

Problemas Identificados/Razões de Falha

O uso do OpenCMS no CI-FCUL não é uma opção válida devido principalmente às seguintes fraquezas:

- A criação de conteúdos não está ao alcance do utilizador comum da aplicação, sendo uma ferramenta orientada para a criação de conteúdos por parte dos programadores e edição simplicada dos mesmos pelos utilizadores.

- Grande dose de programação para implementar qualquer funcionalidade, forma de navegação, ou forma de apresentação de conteúdos, em que essa programação é feita em código JSP e sobre caixas de texto sem qualquer automatismo ou simplificação de processos.

- Ferramenta cada vez menos utilizada em detrimento de outros gestores de conteúdos que permitem extensão de funcionalidades de forma muito mais simplificada.

Magnolia

O que é?

Magnolia, é o nome de um CMS open-source baseado numa interface de repositório de conteúdos para Java, desenvolvido pela Magnolia International Ltd com sede em Basileia, Suíça.

A primeira versão do CMS foi lançada em 2003, tendo sido depois alargada com funcionalidades centradas na usabilidade e relançado em 2004. Em 2006 o Magnolia CMS foi estendido para uma versão enterprise (e por isso utilizável com o pagamento de uma licença de empresa) com funcionalidades alargadas, como suporte para autentificação, gestão de produção e uma ferramenta designada "Sitedesigner" para criação simplificada de conteúdos Web, tendo continuado a ser dado suporte à versão simples de comunidade, sem que estas funcionalidades extra fossem no entanto incluídas.

Características principais para uso no CI

O facto de a versão mais completa deste gestor de conteúdos ser paga, limita a sua utilização no CI-FCUL, no entanto valerá a pena sumarizar as funcionalidades do CMS completo, para perceber a sua perspectiva de utilização.

Criação e Edição de Conteúdo:

Magnolia CMS introduz ao nível criação e edição de conteúdos as funcionalidades típicas de um CMS, possuindo edição centrada no próprio site de gestão Web (Figura 61), os utilizadores podem editar qualquer conteúdo criado com o auxílio de hiperligações de edição que possibilitam acesso a janelas com as diferentes propriedades de cada campo e campos de edição de texto para modificação simplificada.

A criação de novos conteúdos faz-se também de forma dinâmica e relativamente idêntica à edição, acedendo a hiperligações de criação de conteúdos em diferentes espaços no template de apresentação, e através da inserção de texto de forma simplificada nos diferentes campos que definem o corpo do conteúdo. Estas propriedades estão presentes em qualquer uma das versões do CMS, tanto para a Enterprise Edition como para a versão gratuita do mesmo.

Existem ainda funcionalidades de criação de conteúdos apenas incluídas na versão mais completa deste CMS, como é o caso de criação de fóruns, formulários de registo, geração RSS, funcionalidades de agregação de conteúdos variados, etc.

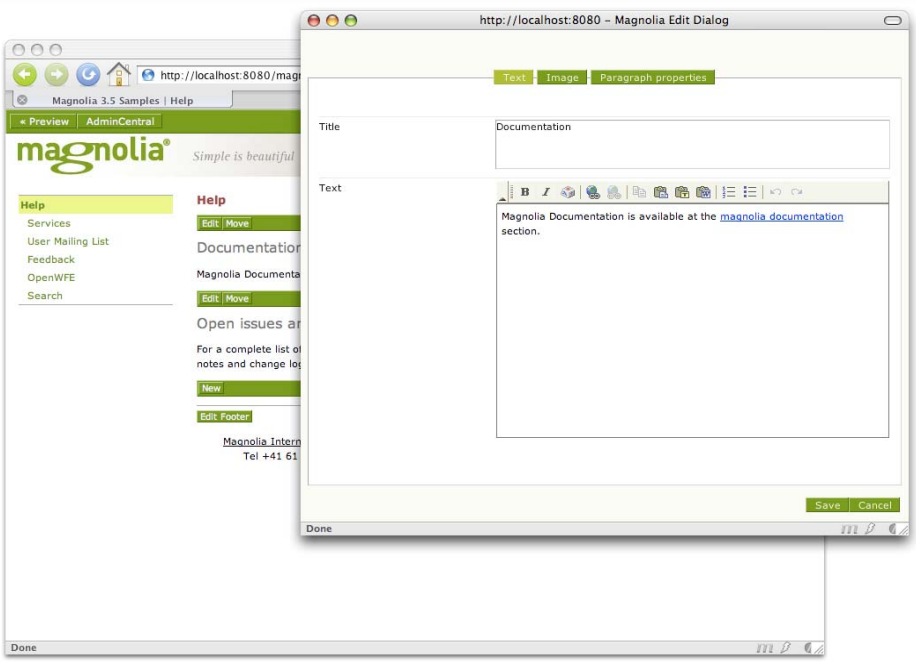


Figura 61: Edição de Conteúdo no Magnólia CMS

Gestão:

Como em quase todos os CMS baseados na produção de conteúdos Web, o Magnólia é também baseado numa aplicação Web que serve de portal de gestão e permite aceder a todos os módulos que o estendem, desde a gestão de permissões de acesso dos diferentes utilizadores, a sistemas de busca de informação optimizados por tags - inseridos aquando da criação dos dados -, suporte para acesso por dispositivos móveis ao próprio portal, sistemas de gestão de todos os ficheiros, imagens e vídeos que fazem parte dos vários sítios que o CMS gere, etc (Figura 62). Todas as funcionalidades estão assim acessíveis através de um sistema de navegação relativamente simples e intuitivo.

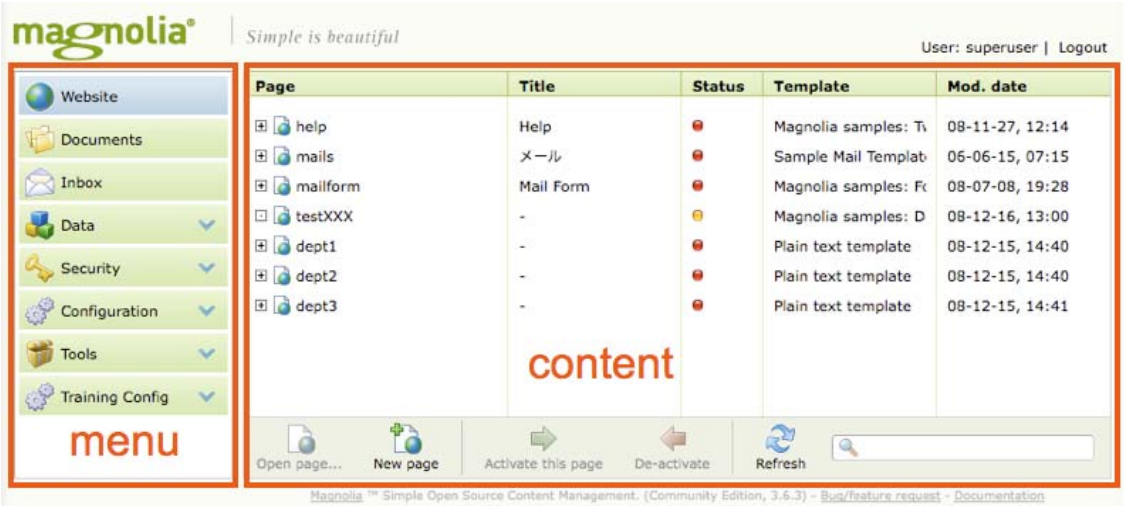


Figura 62: Interface de Administração do Magnólia CMS

Apresentação:

A camada de desenvolvimento de templates, incluída na versão mais completa do CMS, permite aos desenhadores e programadores desenhar o seu sítio Web e controlar as diferentes propriedades do template de apresentação.

A versão comunitária do CMS não contém esta camada, e portanto a definição de um template é feita com a escolha de um já predefinido e programado, sendo que a única possibilidade de configuração se baseia no arrastamento de componentes entre diferentes locais do template, sendo no entanto a definição destes espaços bastante limitada, e apenas ao alcance de programadores mais experientes.

Comunidade e Suporte:

A comunidade de suporte e a existência de guias de desenvolvimento e configuração do Magnolia CMS está igualmente dependente da licença adquirida ou não para uso do software. Com uma licença enterprise, tem-se acesso a diferentes conteúdos exemplificativos de desenvolvimento, guias de programação, módulos extensivos de funcionalidades, bem como suporte directo ao utilizador por parte da empresa responsável. Apenas com uma versão comunitária, existe grande falta de documentos ou guias de suporte, e mesmo os fóruns de comunidade são muito pouco populados levando a uma resolução muito lenta de problemas.

Problemas Identificados/Razões de Falha

O uso do gestor de conteúdos Magnolia no CI-FCUL não é uma opção válida devido à seguinte fraqueza:

- A existência de dois tipos de versões do CMS em que a versão comunitária apresenta um número de funcionalidades muito mais reduzida que a versão completa, representa um grande obstáculo na adopção do CMS. A versão gratuita do programa é claramente insuficiente para o que se necessita no CI-FCUL, enquanto a versão completa apresenta custos demasiado elevados para o Centro, não apresentando funcionalidades a mais que muitos outros CMS gratuitos existentes no mercado.

Plone

O que é?

Plone CMS é um gestor de conteúdos open-source e gratuito construído sobre um servidor aplicacional em linguagem Phyton, baseado também num cliente Web que fornece um ambiente colaborativo com um sítio Web tipo portal.

O projecto Plone iniciou-se em 1999 por Alexander Limi, Alan Runyan, e Vidar Andersen, e pode ser usado tendo em vista vários objectivos, como a implementação de blogs, sítios Web, lojas virtuais e portais internos de empresas, mas essencialmente para ser usado como um sistema de publicação de documentos ou ferramenta de colaboração Web.

As suas principais forças são a sua flexibilidade e capacidade de adaptação de processos, o seu elevado nível de segurança, capacidade de extensão e eficiente usabilidade.

Características principais para uso no CI

Criação e Edição de Conteúdo:

No Plone, é possível a criação de conteúdos por parte dos seus utilizadores através do upload de artigos, ficheiros, eventos de calendário e páginas Web para espaços pessoais ou partilhados por um grupo de utilizadores.

A edição de conteúdos no Plone (Figura 63) para além de poder ser feita com editores de texto simples acessíveis através da interface de administração, pode também ser feita por uma funcionalidade de editor externo que permite aos utilizadores alterarem o conteúdo de uma página utilizando o seu editor de texto favorito e após finalizarem essa edição, submeterem as alterações de volta para o sítio Web.

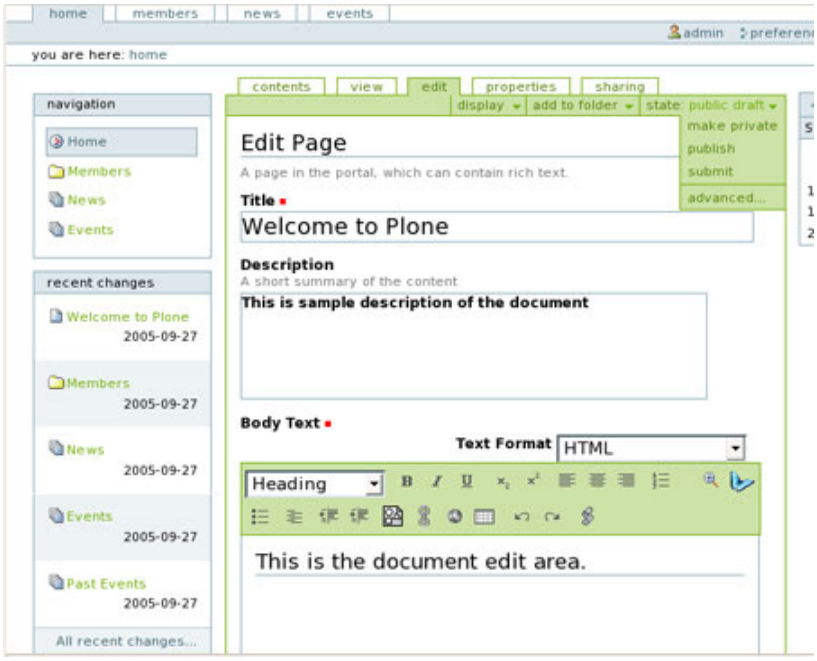


Figura 63: Edição simplificada de conteúdos no Plone

Enquanto algum utilizador faz uso da funcionalidade de edição externa o CMS fornece um sistema de bloqueamento do conteúdo a ser modificado impedindo alterações simultâneas por parte de vários utilizadores.

Para além de conteúdos baseados em ficheiros, no Plone a criação de conteúdo também se faz através da criação por parte dos utilizadores de inteiras páginas Web, itens de notícias e outros tipos de conteúdos como blogs e wikis.

A postagem de comentários nos diferentes conteúdos é uma das tarefas mais repetidas na plataforma, mostrando a forma como este software é principalmente usado como espaço colaborativo ao invés de estar apenas centrado no desenvolvimento de sítios Web.

Gestão:

Sendo baseado numa estrutura de espaços de ficheiros pessoais e de grupo, o Plone possui um mecanismo de controlo de acesso aos diferentes espaços partilhados, configurado pelos proprietários que cada espaço. Assim cada utilizador pode restringir o acesso aos seus conteúdos apenas a certos "colegas".

A interface de administração do Plone (Figura 64) é baseada numa página Web em que cada aba diz respeito a uma funcionalidade ou propriedade do gestor de conteúdos, existindo diferentes áreas para utilizadores, eventos e notícias tal como para historial de acções realizadas, partilha de ficheiros e sistemas de regras sobre os conteúdos.

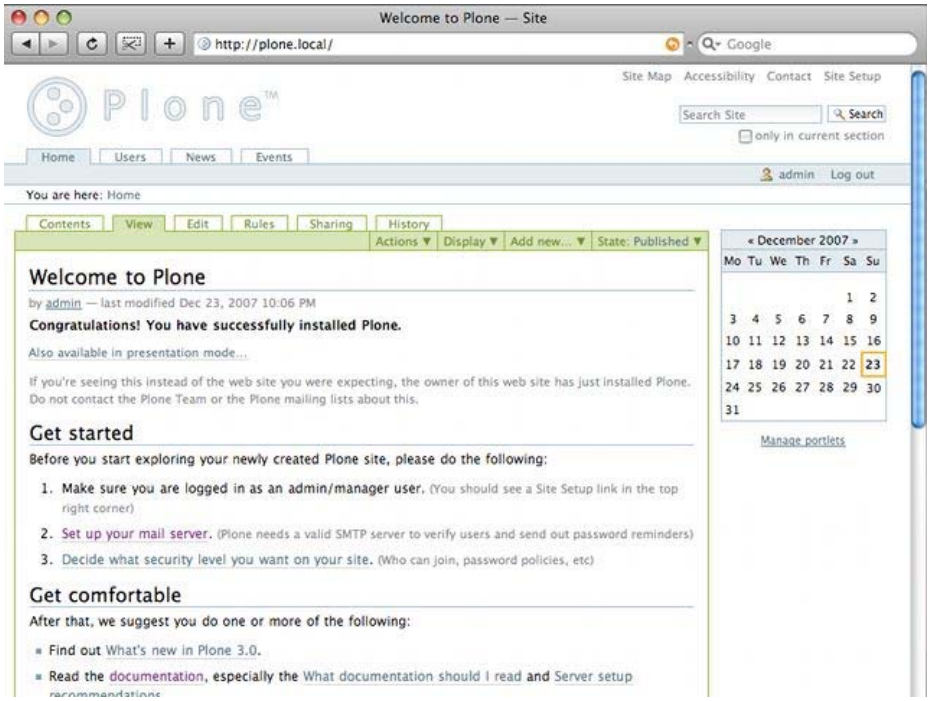


Figura 64: Interface de Administração do Plone com as referidas abas para as diferentes funcionalidades

Através de um módulo instalável, este CMS pode também oferecer funcionalidades de versionamento de ficheiros de forma automática.

Apresentação:

A interface de administração do Plone, permite administrar o seu aspecto de apresentação e definir e inserir no seu sítio Web: imagens, fontes, cores e fluxos de configuração dos ficheiros carregados pelos utilizadores.

Modificações mais profundas sobre o template têm sempre de ser efectuadas através de programação Python, através de uma linguagem de programação própria do Plone bem como ainda através de configuração de ficheiros CSS.

O sistema de apresentação do conhecido site MediaWiki's denominado "Monobook" é um exemplo de um layout definido pelas folhas de estilo do Plone.

Comunidade e Suporte:

Plone é uma das ferramentas de gestão de conteúdos mais conhecidas no meio informático e por isso conta com uma comunidade extensa e rica, possibilitando a existência de muitos fóruns de dúvidas, desenvolvimento suportado, guias de programação, bem como a existência de grande número de extensões desenvolvidas pelos próprios utilizadores sobre a forma de produtos que são depois distribuídos de forma livre no sítio do CMS.

A existência de livros dedicados apenas a programação e uso desta ferramenta e de mailing lists de esclarecimento de dúvidas que funcionam de uma forma muito eficaz, fundamentam de sobremaneira a utilização eficaz do CMS Plone.

Problemas Identificados/Razões de Falha

O uso do CMS Plone no CI-FCUL não é uma opção válida devido às seguintes fraquezas:

- Necessidade de conhecimento da linguagem Python para alterações relacionadas com a forma de apresentação dos conteúdos, e o facto de esta linguagem ser de conhecimento menos comum que a linguagem Java ou a linguagem PHP, tornam-no um candidato menor para adopção no Centro. Junta-se a estes factos o facto de a posterior adaptação com a plataforma de desenvolvimento desenvolvida sobre Java adivinhar-se muito difícil.

- Utilização do CMS centrada na gestão de documentos e na concretização de um espaço colaborativo Web e não no desenvolvimento ou gestão de conteúdos estáticos de sítios Web.

Concrete5

O que é?

Concrete5 é um CMS mais simples que a maioria, que pretende através de poucos e simples passos objectivar a criação de um sítio de conteúdos também eles relativamente simples e exclusivamente estáticos. Como gestor de conteúdos Web, permite a edição de conteúdos directamente na página, sem contar com grande espaço de administração ou configuração "BackOffice". Auto intitula-se como "O CMS feito para o marketing mas construído para os entendidos", e baseia-se inteiramente em linguagem PHP 4. Sobre os objectivos de facilitar, flexibilizar e escalar a construção de sítios Web simples, foi criado originalmente por Franz Maruna e Andrew Embler e lançado na sua primeira versão em 2003.

Características principais para uso no CI

Criação e Edição de Conteúdo:

Uma das funcionalidades mais marcantes no Concrete5 é a forma facilitada de edição de conteúdos Web. Em modo de edição todos os conteúdos se transformam em "blocos" e aparecem delimitadas por tracejados que os identificam. Clicar sobre cada conteúdo mostra-nos as opções de edição, desde alterar o texto a mover de sítio o bloco no template de apresentação (Figura 65).

A edição de outros tipos de conteúdos também está presente e facilitada pela interface do CMS. Inserir vídeos, mapas GoogleMaps, "slide shows" de fotografias e feeds RSS, são funcionalidades tidas como certas no Concrete5.

Figura 65: Edição de conteúdos no Concrete5

Uma das principais falhas deste CMS ao nível de criação e gestão de conteúdos é o facto de possibilitar a criação de muito poucos tipos de conteúdos, isto é, é muito difícil por exemplo agregar numa única página vários conteúdos dispersos pelo sítio Web (fazer uma página de notícias por exemplo) ou criar conteúdos compostos por vários tipos de informação.

Gestão:

As principais funcionalidades precisas numa base regular de edição e gestão de conteúdos estão contidas numa barra de ferramentas sempre visível aquando da edição de qualquer pedaço de informação em páginas individuais. Quanto às restantes funcionalidades estão acessíveis numa espécie de pequeno "backend" acessível para configuração de temas e processos de apresentação, obtenção de relatórios, e execução de outras tarefas administrativas (Figura 66).

No que diz respeito às funcionalidades que este CMS oferece, destacam-se: sistema de versionamento e de captação de alterações feitas nos conteúdos, capacidade de pré-visualização das alterações antes de estas serem de facto submetidas, gestão de utilizadores e seus papéis e permissões ao nível da aplicação e dos conteúdos, ferramenta de reaproveitamento de conteúdos do sítio Web, corrector ortográfico, sistema de URLs optimizado para sistemas de busca, integração com ferramentas Google, calendarização de tarefas sobre conteúdos, e outras funcionalidades menores.

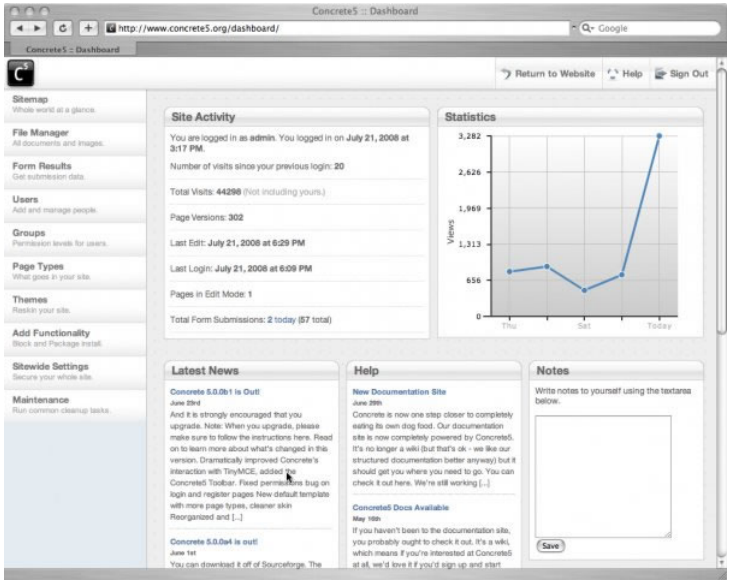


Figura 66: Interface de Administração do CMS Concrete5

Apresentação:

O sistema de apresentação de informação implementado pelo Concrete5 permite antes de mais uma configuração de temas por tipos de páginas, sendo que cada tema pode assim definir vários tipos de páginas. Esta é uma das suas principais características, mas também uma das suas principais falhas,

sabendo-se possível o arrastamento de conteúdos pelas diversas partes que definem o template é no entanto de difícil concretização a definição de pequenas alterações sobre cada template de apresentação.

Alterações são apenas feitas por programadores não estando ao alcance de simples utilizadores a execução de pequenas alterações na apresentação da informação (apenas são livres de modificar o local de apresentação de determinado conteúdo informativo).

Outra das grandes particularidades ao nível dos templates de apresentação, é o facto de estes estarem divididos pelos diferentes locais de uma página, isto é, para o cabeçalho usa-se um template, para o rodapé usa-se outro, tal como para a página principal, para as colunas laterais e para as colunas verticais. Cada componente de uma página apresenta um template de apresentação, dividindo uma página num número demasiado incomodativo de templates ao invés de definir apenas um template composto por partes específicas. Esta particularidade dificulta de sobremaneira a gestão da apresentação de um sítio Web.

Comunidade e Suporte:

Em termos de comunidade que faz uso desta plataforma CMS, existe um número relativamente reduzido de organizações e utilizadores activos quando em comparação com outros CMS (Drupal ou Joomla por exemplo). Isto deve-se ao facto de não haver total abertura da plataforma em relação a contribuição externa dos seus utilizadores. Novas actualizações e novas funcionalidades são sempre introduzidas pelos próprios desenvolvedores do CMS, perdendo-se grandes oportunidades de desenvolvimento conjunto de novos tipos de funcionalidades o que resulta num grau bastante elevado de abandono da aplicação por parte dos seus utilizadores mais experientes e numa evolução demasiado lenta de novas funcionalidades.

Problemas Identificados/Razões de Falha

Em termos de instalação do produto, verificaram-se algumas dificuldades devido a uma grande intransigência nas especificações de instalação de certos módulos dispensáveis ao funcionamento geral. Assim que se procede à instalação do software, é necessário passar por um elevado número de configurações PHP, que não se justificam dado a simplicidade das tarefas realizáveis por esta plataforma. Para além disto, qualquer tentativa de instalação que se desvie daquela que é aconselhada resulta num bloqueio da mesma e na publicitação de uma instalação profissional paga do mesmo programa.

No entanto, e em termos gerais, Concrete5 é uma ferramenta de gestão de conteúdos Web muito promissora e que desempenha funções de edição e gestão de informação de forma muito eficaz, apresentando uma interface de utilização muito rápida.

Os principais obstáculos para a sua adopção no Centro de Informática da FCUL são então os seguintes:

- Criação de diferentes tipos de conteúdos é escassa, sendo apenas funcionalidade regular, a criação de conteúdos simples de texto, embora mediante instalações (nada simples) modulares, seja possível criar outro tipo de conteúdos (sendo mesmo assim o leque de opções muito escasso).

- Poucas possibilidades de personalização, configuração e gestão ao nível dos templates de apresentação. Alterações de maior relevo que alterar blocos de informação de sítios, estão apenas ao alcance de programadores mais experientes. Existência de vários templates, uma para cada zona do ecrã, dificuldade a gestão centralizada da apresentação.

- Uso pouco extensivo do Gestor de Conteúdos ao nível de gestão variada, sendo necessária a sua instalação um número de vezes igual ao número de Sítios Web que se pretende gerir com a ferramenta. Por cada nova aplicação que se pretende gerir, uma nova instalação do CMS tem que ser feita.

- Gestão de actualizações ao nível do CMS pode-se tornar quase impossível de gerir devido ao elevado número de instalações necessárias aquando da existência de um grande número de sítios Web.

Umbraco

O que é?

Como apresentação, o Umbraco é um CMS que pretende ser simples, óbvio e amigável para os seus utilizadores, e centrado na edição e criação de conteúdo. Desenhado com os padrões de interacção mais conhecidos que existem (aqueles que estamos habituados a encontrar nos programas mais usados) mas que permite aos seus utilizadores grande número de funcionalidades como funções de segurança baseada em tarefas, editores de texto WYSIWYG tipo Microsoft Word, gestão de ficheiros, sistemas de notificação de alterações, etc.

Ao contrário de todos os outros Gestores de Conteúdos apresentados neste relatório, Umbraco é um CMS baseado em programação. NET e não nas típicas linguagens Java e PHP, mas que faz sentido referir neste estudo, uma vez que desempenha de forma exemplar as tarefas necessárias no CI-FCUL e que pode por isso servir de base de comparação para com todos os outros.

O conceito tido no .NET de áreas de conteúdo ("contentPlaceHolder") carregadas em partes específicas de todo o template de apresentação ("masterpage"), seria o conceito ideal de implementação no Centro de Informática, porque permitiria separar de forma clara a edição de "pacotes de conteúdo" da edição de "pacotes de apresentação", não tendo o desenhador que se preocupar com o impacto de uma actualização ou modificação de conteúdos na apresentação geral do sítio Web, nem o editor que se preocupar com a forma como adicionar conteúdo poderia gerar erros de apresentação no sítio geral.

Características principais para uso no CI

Criação e Edição de Conteúdo:

No Umbraco as páginas de conteúdo são estruturadas e apresentadas sobre a forma de uma árvore de ficheiros, tal como acontece normalmente no explorador de ficheiros do Microsoft Windows.

Para criação de páginas basta clicar com o botão direito no local pretendido na árvore de conteúdos e activar o botão de criação que está em todas as alturas visível, para editar basta seleccionar a página de conteúdos que se pretende alterar, para gravar conteúdos e para publicar conteúdos basta pressionar também os botões respectivos (Figura 67).

Existe uma total compatibilidade de integração na edição de conteúdos usando processadores de texto bem conhecidos como o Microsoft Word.

Neste gestor de conteúdos é possível a criação e edição de conteúdos com base em linhas temporais e portanto servindo-se de calendários os utilizadores podem "programar" a criação ou edição de conteúdos.

Ao nível dos tipos de conteúdos possíveis de criar nesta ferramenta interessa referir que devido ao vasto uso de bibliotecas .NET, é possível criar quase qualquer tipo de conteúdo que o utilizador pretenda e apresentá-lo devidamente no sítio Web.

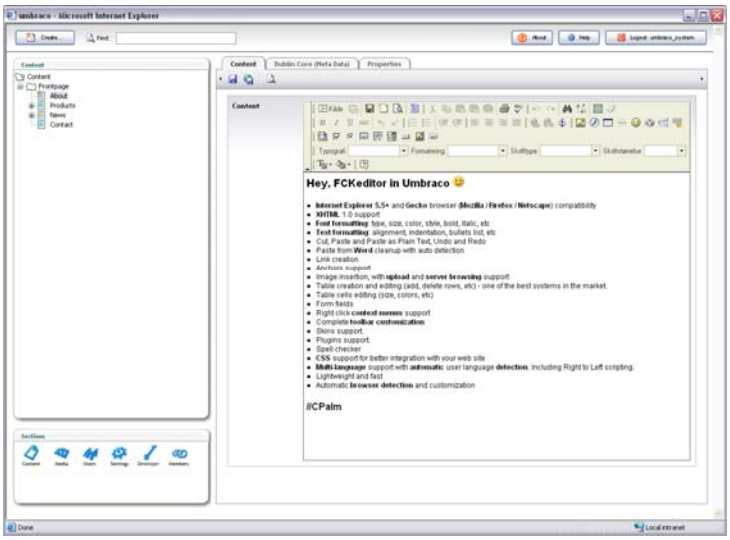


Figura 67: Edição de conteúdos no Umbraco

Gestão:

A primeira funcionalidade evidente em termos de gestão de conteúdo é a de estruturação e gestão de ficheiros através de um explorador de dados que permite as operações mais básicas de gestão (que já foi referido anteriormente) sobre qualquer tipo de ficheiro de dados, desde páginas Web a imagens e vídeos.

Para além disto, Umbraco CMS oferece funcionalidades de versionamento de conteúdo, com uma gestão semi-automática de alterações simultâneas ou sequenciais sobre os mesmos dados por vários utilizadores bem como voltar a versões anteriores dos vários ficheiros ignorando alterações mais recentes caso seja necessário.

Juntamente com a gestão de versões, este CMS fornece também funcionalidades de fluxos de tarefas (workflows) bem como de notificação, que permite implementar ao nível da gestão de conteúdos uma rigorosa rede de heurísticas (Figura 68).

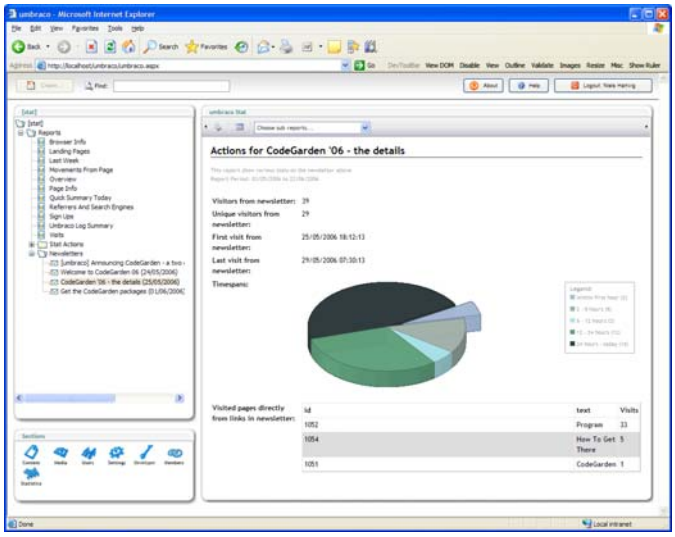


Figura 68: Acesso a uma funcionalidade na interface de Administração Umbraco

Apresentação:

Umbraco suporta várias versões de apresentação de conteúdos em diferentes línguas. Esta funcionalidade é possibilitada através da integração com serviços de tradução fazendo uso de ferramentas de exportação e importação de conteúdos por XML.

A "construção da apresentação" neste CMS é uma característica forte, utilizando linguagem de markup (HTML) que mais lhe convém, o programador constrói o template que pretende, podendo depois reutilizá-lo ou não para outros sítios Web. A configuração do template é uma tarefa de programação e não uma tarefa de escolha de opções, tornando qualquer tarefa relacionada com a apresentação de conteúdos uma tarefa inacessível ao simples utilizador da aplicação (podendo-se tornar em certos casos um factor negativo).

Após construção de um template de apresentação, é possível depois definir quais os diferentes espaços de apresentação de conteúdos ("Placeholders") e ligá-los aos diferentes conteúdos a serem apresentados (permitindo a tal separação entre edição e apresentação de conteúdos).

Comunidade e Suporte:

Uma vantagem evidente de suporte sobre esta ferramenta de gestão de conteúdos, é o facto de esta ser compatível com qualquer Sistema Operativo existente, bem como com qualquer browser Web mais moderno (firefox, internet Explorer, chrome, safari…).

Para além de simplicidade ao nível de compatibilidade, Umbraco CMS apresenta uma forte comunidade de utilizadores e programadores que forma uma forte rede de apoio ao desenvolvimento de Sítios Web. A existência de fóruns, tutoriais e livros no sítio oficial do CMS bem como em outros sítios não oficiais, forma uma forte "família" de suporte para a aplicação.

Apenas de referir, como ponto adicional, a existência ainda de um suporte mais personalizado por parte da equipa de desenvolvedores a utilizadores pagantes de uma licença profissional do CMS.

Problemas Identificados/Razões de Falha

Apesar de grande facilidade de uso e de ser um CMS com todas as características desejadas, Umbraco não poderá ser adoptado pelo CI-FCUL no sentido em que é programado sobre .NET, não sendo compatível nem com Java nem com PHP, tornando-se impraticável para uma possível integração com outras ferramentas de desenvolvimento a serem utilizadas programadas sobre linguagens Java e PHP. No entanto, e como já referir anteriormente, a presença desta CMS neste documento é importante no sentido em que constitui uma preciosa base de comparação para com todos os outros gestores de conteúdos abordados.

eZ Publish

O que é?

eZ Publish ou em português "Fácil Publicação" é um sistema de gestão de conteúdos open-source desenvolvido por uma companhia Norueguesa denominada ez Systems. Escrito em linguagem PHP e XML (para separação do conteúdo e da apresentação), correndo sobre um servidor Web Apache, e permitindo o uso de quase todo o tipo de bases de dados (MySQL, PostgreSQL, SQLServer e Oracle), é de uso e descarregamento gratuito via Web e a empresa fornece também suporte sobre a aplicação aos seus utilizadores, sendo no entanto este suporte, devidamente pago (o único suporte não pago é aquele fornecido pela comunidade de utilizadores).

No horizonte de utilização desta aplicação interessa referir a sua principal utilidade no rápido desenvolvimento e costumização de aplicações Web (desde páginas pessoais a sítios de empresas), com funcionalidades referentes a funções de comércio electrónico, versionamento de conteúdo, gestão de perfis de utilizador, e procura e impressão de conteúdo. Companhias como o MIT, revista Vogue e a Marinha Americana, utilizam este CMS.

Características principais para uso no CI

Criação e Edição de Conteúdo:

Este cms é gerido inteiramente através de um Browser Web e constituído por um editor rico de texto que permite formatação de conteúdos de forma semelhante a um processador de texto, permitindo independência de técnicas HTML (Figura 69). Para além deste tipo de edição directa, é permitida também a importação de ficheiros de texto escritos em editores de texto como forma de edição de conteúdos online.

Figura 69: Publicação de conteúdos através da barra de ferramentas de edição simples

No CMS eZPublish todo o conteúdo pode ser traduzido para várias linguagens, sendo esta administração feita exclusivamente através de um único sítio Web. A selecção de uma linguagem por parte de um utilizador resulta na tradução automática do conteúdo para essa mesma língua. A edição e criação de conteúdos nas diferentes linguagens é realizada pelos respectivos editores e a criação de diferentes URLs para acesso às diferentes páginas de conteúdo do sítio é uma realidade neste gestor de conteúdos.

Uma das actividades mais focadas deste gestor de conteúdo é a de criação e edição de galerias de imagens com carregamento simultâneo de um grande conjunto de imagens que integram essas mesmas galerias. Através de poucas instruções, o eZPublish cria e gere conteúdo deste género num sítio Web.

Tal como para as fotografias, este CMS também se revela muito simplificativo no processo de carregamento de vídeos e ficheiros para um sítio Web, servindo tanto para armazenamento de ficheiros deste tipo, como para transmissão deste tipo de conteúdos.

A pesquisa de conteúdos directamente no sítio Web também é possível através deste CMS, utilizando um sistema simplificado baseado em procuras por tags e nomes de conteúdos, havendo ainda um módulo próprio (eZFind) que estende esta propriedade.

Gestão:

Como já foi referido anteriormente, toda a gestão desta plataforma de conteúdos Web é baseada no uso de um sítio Web que permite o acesso a todas as configurações relacionadas com o conteúdo criado e com as diferentes funcionalidades. Tornando-se assim acessível através de qualquer sítio que se encontre ligado à Internet.

EZ Publish em termos de tarefas de administração (Figura 70) é bastante completo apresentando uma interface de administração com funcionalidades de controlo e gestão de acessos e worklows, gestão de versões de conteúdos, gestão de produção e optimização do sistema de procura de ficheiros.

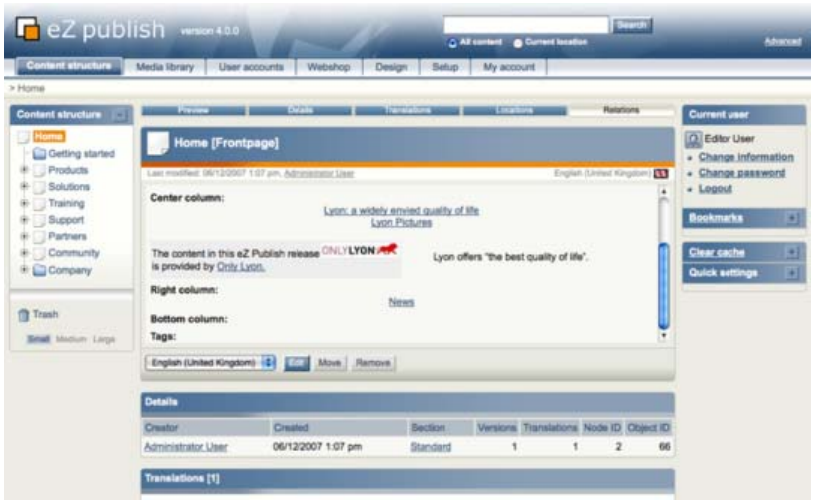


Figura 70: Interface de gestão de conteúdos eZ Publish

A gestão de acessos Ez Publish baseia-se na existência de utilizadores e de papéis possíveis de atribuir aos vários utilizadores. Cada papel reflecte um número e tipo de permissões específicas sobre os conteúdos e funcionalidades da plataforma, permitindo ou bloqueando possibilidades de interacção aos vários membros.

É ainda possível neste CMS definir uma sequência de acções necessárias para que um conteúdo seja alterado, publicado, apagado, etc., envolvendo permissões e diferentes papéis de utilizadores.

À semelhança dos workflows, também a gestão das versões dos diferentes ficheiros pode ser mecanizada na própria aplicação, facilitando a consulta de alterações que cada ficheiro sofreu ao longo do tempo.

Apresentação:

Templates de apresentação para este CMS são escritos usando uma linguagem de scripting própria da plataforma, bem como definindo CSSs para os estilos pretendidos. Essa linguagem é baseada em HTML e na inserção de marcas que representam a lógica de apresentação, constituindo uma linguagem relativamente fácil de aprender.

Através da interface de administração é possível configurar alguns aspectos da apresentação de todo o sítio Web, aspectos como o estilo de navegação, as cores, a fonte da letra, são disso exemplo.

Finalmente no que ao uso de URLS diz respeito, qualquer página de conteúdo está acessível através de uma morada própria e o uso de técnicas de “clean url” e de “aliases” possibilita uma grande flexibilidade no acesso aos diversos conteúdos.

Comunidade e Suporte:

As principais funcionalidades desta plataforma baseiam-se em componentes. “eZ Components” é uma biblioteca de módulos definidos para acelerar o desenvolvimento de aplicações, incluindo funções relacionadas com a performance (caching), conexão com bases de dados, debugging, RSS e ferramentas de análise de tráfego, conversão de formatos, suporte para correio electrónico e sistemas de validação de input.

A existência de algumas comunidades activas de utilizadores deste gestor de conteúdos Web, resulta na existência de diversas extensões com diversas funções individuais (como criação de funcionalidades de blog, tags, fóruns, sistemas de votação, etc.) permitindo uma actualização e renovação de componentes enquanto se preserva a compatibilidade entre as várias partes.

No entanto, quase todos os módulos que procuram implementar novas funcionalidades neste CMS não são de código aberto como na maior parte dos outros gestores de conteúdos, limitando a personalização da plataforma por parte dos seus utilizadores.

Para além de comunidades “amadoras” existe ainda uma rede de parceiros certificados que realizam implementações ao nível da plataforma. No momento serão cerca de 200 parceiros por todo o mundo com destaque na Noruega, Ucrânia, Franca, Canadá e Alemanha.

Problemas Identificados/Razões de Falha

O uso do CMS EzPublish no CI-FCUL não é uma opção válida devido às seguintes fraquezas:

- EzPublish não apresenta um funcionamento tão facilmente perceptível quanto outros CMS abordados, e mesmo após habituação, a eficiência resultante das suas operações é relativamente reduzida, não se tornando num sistema de gestão de conteúdos em que o elevado uso permite uma alta eficiência nas operações.

- Uso pouco generalizado por todo o mundo, e o facto da comunidade não poder contribuir de uma forma mais activa na modificação de módulos existentes limita a evolução do sistema, limitando consequentemente o seu uso.

- Uso pouco extensivo do Gestor de Conteúdos ao nível de gestão variada, sendo necessária a sua instalação um número de vezes igual ao número de Sítios Web que se pretende gerir com a ferramenta. Por cada nova aplicação que se pretende gerir, uma nova instalação do CMS tem que ser feita.

- Gestão de actualizações ao nível do CMS pode-se tornar quase impossível de gerir devido ao elevado número de instalações necessárias aquando da existência de um grande número de sítios Web. Isto deve-se devido ao facto de ser necessário instalar a aplicação uma vez para cada sítio Web que se pretende criar ou gerir.

- Existência de funcionalidades pagas limita a extensão da plataforma de forma generalizada.

Jahia

O que é?

Jahia CMS é uma plataforma gestora de conteúdos Web que oferece também funcionalidades típicas de um gestor de ficheiros ou de um Portal de serviços. Com lançamento muito recente e programado sobre linguagem Java, este gestor de conteúdos tem um funcionamento muito intuitivo e baseando-se numa aplicação Web apresenta grande capacidade de integração entre os seus vários módulos, sendo possível por exemplo integrar a exploração de uma pasta de ficheiros numa página Web através de um sistema apresentação de conteúdos escrito em Java.

O utilizador desta plataforma não precisa de ter quaisquer fundamentos de programação em Java para proceder à sua utilização, sendo o próprio processo de experimentação da aplicação suficiente para que em pouco tempo, este tirar total partido do sistema.

Características principais para uso no CI

Criação e Edição de Conteúdo:

A criação de conteúdos no Jahia é feita com o auxílio de pequenos botões fixados na própria interface do sítio Web, que permitem, em modo de edição, escrever conteúdo de forma simplificada e sobre um editor WYSIWYG (Figura 71). A conciliação simultânea com código HTML permite a edição de outros tipos de conteúdo para além do típico texto, com importação de imagens, vídeos embebidos, etc.

O desenvolvimento por parte de programadores de conteúdos especiais, como por exemplo inteiras aplicações, e sua consequente integração no sítio Web é feita à semelhança da criação de conteúdo, sendo possível um utilizador inserir num espaço de apresentação o dito "pacote de aplicação" e desta forma implementar uma nova funcionalidade directamente no sítio Web.

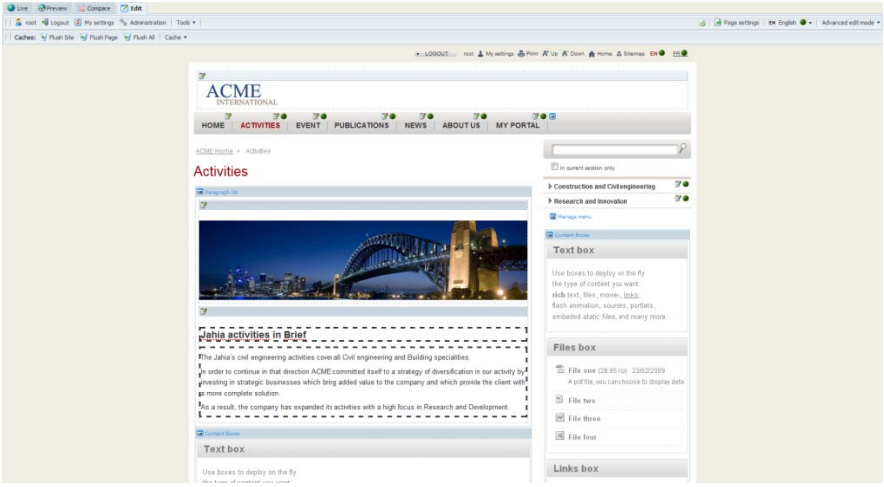


Figura 71: Edição de conteúdos no Jahia CMS

Gestão:

No Jahia a gestão multi-sítio é uma realidade, sendo sempre possível a qualquer altura, saltar entre diferentes serviços, ou partir para a criação de novos. Através da interface de administração (Figura 72) todos os sítios Web são geridos. Co-existindo na mesma aplicação vários serviços diferentes, é também fornecido um sistema de procura de informação que permite percorrer todos os conteúdos de todos esses sítios e identificar em qual deles se encontra a informação desejada.

Jahia oferece também total compatibilidade com outras línguas de apresentação e de interacção, sendo por isso mais flexível a sua utilização em diferentes partes do mundo. A própria empresa responsável pelo desenvolvimento do CMS oferece suporte nessa área, a utilizadores registados na plataforma.

Como gestor de ficheiros que também é, esta ferramenta oferece um simplificado explorador de dados que permite efectuar sobre os vários documentos as operações básicas de uma ferramenta do género.

A existência de um sistema de regras e fluxos de publicação que abrange tarefas e os diferentes utilizadores da aplicação aquando da publicação de um novo conteúdo, permite haver total controlo sobre a fidelidade da informação contida no sítio Web, e faz com que todos os utilizadores saibam aquilo que se passa ao nível das actualizações de conteúdos.

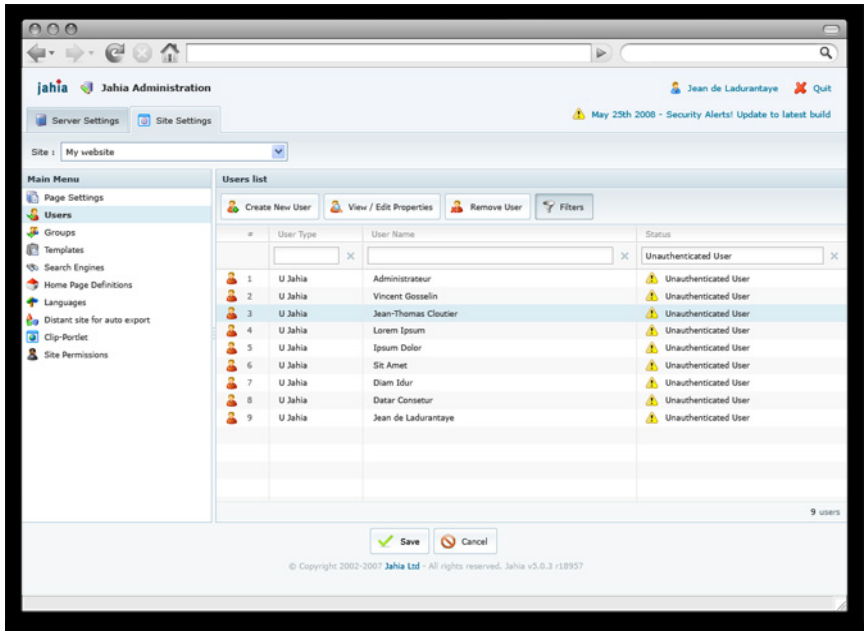


Figura 72: Interface de Administração do Jahia

Apresentação:

Ao nível da forma como todos os conteúdos são apresentados nesta plataforma, tem-se a existência de um mecanismo de templates baseados também eles em Java (JSP), que não possibilitam ao utilizador comum, muito mais que a escolha de um modelo que esteja de acordo com as pretensões para o sítio Web. Tarefas de personalização desse template, como escolha de cor, de fontes, etc., não são fornecidos

ao nível do CMS, e a única forma de os modificar é mesmo por via programática, sendo necessário para isso, conhecimentos de programação relativamente avançados. Também ao nível da comunidade, o número de templates possíveis de instalar são muito poucos (em parte devido ao facto deste CMS ser relativamente recente), mas uma vez que todo e qualquer template é apenas código, a adaptação de templates usados em outras ferramentas é sempre a possibilidade mais apetecível.

Ao nível do isolamento das várias páginas de conteúdos, existe uma grande falha ao nível do Jahia CMS. Qualquer conteúdo editável de forma simplificada no editor online, na realidade não está visível de uma forma isolada no código fonte da página em causa, e portanto, numa perspectiva de utilização e actualização automática de conteúdo deste CMS noutra plataforma de desenvolvimento, existem grandes limitações. A própria morada do sítio Web gerado é fixa independente do conteúdo que está a ser mostrado, dando a ideia que a apresentação do conteúdo é feita de alguma forma "por cima" do template de apresentação, ficando visível para o utilizador mas praticamente invisível programaticamente para o desenvolvedor de conteúdos.

Comunidade e Suporte:

A principal fraqueza neste sistema de gestão de conteúdos encontra-se ao nível da comunidade que lhe serve de suporte. Sendo uma ferramenta muito recente não conta ainda com grande número de utilizadores, e mesmo o próprio sítio Web esta vagamente desenvolvido no que a fóruns ou formas de discussão diz respeito. Até a raridade com que se encontram resultados relativos a este CMS com uma normal pesquisa no Google é na realidade assustadora, e a consulta atenta aos raros documentos encontrados, dá a entender uma versão mais antiga da aplicação totalmente diferente em comportamento da actual. Esta disparidade nos documentos e nas formas de suporte também não ajudam a "sustentar" o uso ou adopção do Jahia como ferramenta gestora de conteúdos.

A princípio a existência de alguns guias centrados no desenvolvimento de aplicações modulares capazes de serem integradas na ferramenta principal deixariam antever boas perspectivas para o futuro deste CMS, no entanto no presente, a total ausência de qualquer módulo de novas funcionalidades para possível integração prova a existência de uma "falésia" prática entre esses documentos teóricos e a sua aplicação real, não sendo bom pronuncio de futuro para este CMS.

Problemas Identificados/Razões de Falha

O uso do Jahia CMS no CI-FCUL não é uma opção válida devido às seguintes fraquezas:

- Sendo um CMS muito recente e praticamente desconhecimento por todo o mundo e a sua comunidade de utilizadores é praticamente inexistente. Com todas as dificuldades que dai advêm, o suporte ao desenvolvimento, bem como a procura por documentação, bem como ainda o descarregamento de módulos que implementam funcionalidades, é demasiado escasso para uma ferramenta deste género.

- Grande falta de personalização e de simplicidade de implementação ao nível dos templates de apresentação, sendo a criação destes um puro acto de programação, tal como qualquer alteração ou

personalização, uma tarefa apenas ao alcance de programadores mais experientes e familiarizados com linguagem JSP.

- Facto de todo o sítio Web resultante da plataforma ter sempre o mesmo URL independentemente da página de conteúdos que está a ser mostrada, bem como o facto de os conteúdos não aparecerem devidamente isolados ao nível do código da página, impossibilita qualquer forma de integração dos conteúdos estáticos gerados pelo CMS com uma plataforma de desenvolvimento mais dinâmica.

Joomla

O que é?

Joomla é um Sistema de Gestão de Conteúdos Web desenvolvido e originalmente distribuído por uma firma australiana denominada Miro International, que fornece ferramentas de publicação e colaboração na Web.

A arquitectura do Joomla é baseada em PHP e MYSQL e tem como principal objectivo a grande facilidade de uso. Isto é conseguido através de uma interface de utilizador que corre sobre em browser e que se caracteriza por ser muito simples e por permitir, através de simples passos, uma gestão eficaz de sítios Web. O Joomla é muito fácil de instalar, configurar e implementar, principalmente se for apenas baseado nas formas de apresentação que vêm incluídas no pacote de instalação.

Características principais para uso no CI

Criação e Edição de Conteúdo:

A criação de páginas de conteúdos estáticos, ou itens de conteúdos (que agregam diferentes conteúdos estáticos) no Joomla é muito simples e baseada num editor de texto WYSIWYG denominada tinyMCE (Figura 73).

A edição de imagens juntamente com texto é facilitada pelo uso da interface de edição que permite agregar imagens a conteúdos sem utilização de referências HTML ajudando a que as referências nunca sejam perdidas.

Para além de edição possibilitada no próprio sítio Web fazendo uso das ferramentas de edição directa, um utilizador pode também realizar alterações aos conteúdos através da interface de administração. No entanto as alterações através desta interface estão normalmente reservadas a utilizadores mais experientes de forma a limitar o acesso a opções de edição mais avançadas a estes.

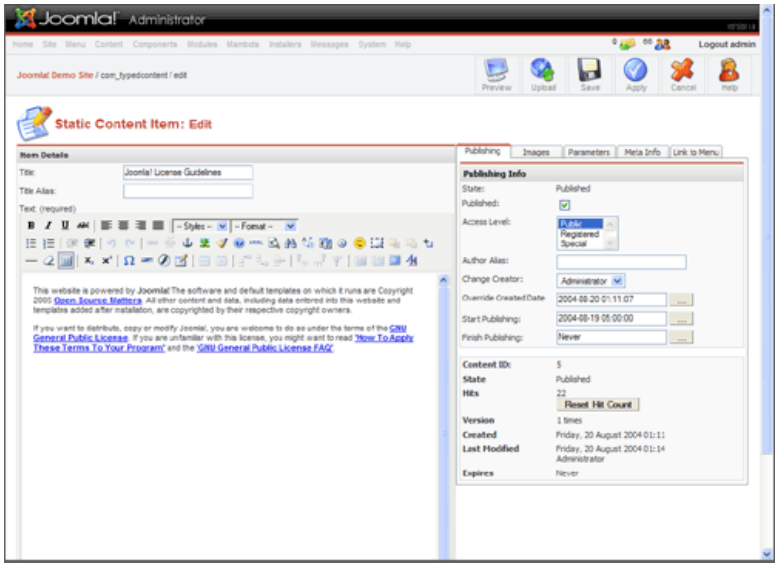


Figura 73: Edição de conteúdo no Joomla

Gestão:

Joomla define estados de publicação para os seus conteúdos, podendo existir conteúdos por publicar e conteúdos publicados.

A ausência de um sistema de gestão de versões faz com que qualquer alteração publicada ao nível de uma página se torne imediatamente visível para todos os utilizadores. No entanto, aquando de uma alteração o autor tem sempre a hipótese de visualizar o resultado das alterações antes de as publicar, evitando-se erros em actualizações de conteúdos.

Joomla apresenta um sistema de bloqueamento de conteúdos que evita que vários utilizadores alterem o mesmo conteúdo de forma simultânea, evitando problemas derivados de alterações concorrentes aos conteúdos.

Através das interfaces de gestão e administração do Joomla (Figura 74), conteúdos são inseridos em páginas de forma associativa, tal como são inseridos como ligações em menus ou adicionados a listas de conteúdos ou itens de navegação. Desta forma se insere e se liga conteúdos por todo o sítio Web construído.

A possibilidade de gestão simultânea de vários sítios Web através da mesma interface de administração é uma das características mais fortes deste CMS, uma vez que centra numa única versão da plataforma a gestão simultânea de uma grande número de sites.

Joomla possibilita ainda a integração de outras funcionalidades através da instalação de módulos gratuitos, como é o caso de um serviço de pesquisa interna de conteúdos, uma funcionalidade “cesto de compras”, uma funcionalidade que implementa galerias fotográficas, etc.

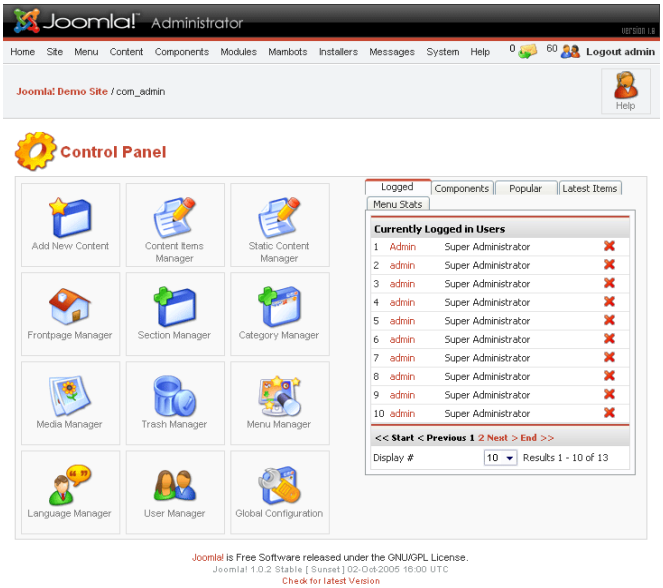


Figura 74: Interface de Administração do Joomla

Apresentação:

A forma de apresentação dos diferentes conteúdos no Joomla é baseada em templates e skins. Templates definem a apresentação geral, enquanto as skins definem variantes ao template, como por exemplo, cores, fontes, localização de blocos de conteúdos, etc.

A instalação de templates e skins é feita através da interface administrativa, e em qualquer altura é sempre possível alterar os mesmos acedendo ao código PHP e fazendo alterações no mesmo.

Skins possibilitam alterações possíveis de fazer por utilizadores comuns, enquanto o código PHP possibilita alterações a utilizadores mais experientes e conhecedores de programação.

Comunidade e Suporte:

A comunidade Joomla através do seu site oficial, possibilita o descarregamento de novas funcionalidades desenvolvidas no dia-a-dia por utilizadores/programadores, tal como possibilita o acesso às mais variadas formas de apresentação (templates e skins) e formas de navegação (menus).

Este CMS apresenta uma comunidade muito forte, sendo um dos gestores de conteúdos Web mais utilizados por todo o Mundo e por isso contando com grande número de sítios Web que oferecem extensões de funcionalidades, permitem resolução de problemas, sugerem tutoriais de treino e possibilitam ao utilizador manter-se sempre informado sobre as últimas novidades na tecnologia Joomla (fóruns registam uma actividade de mais de 1000 postagens por dia).

Problemas Identificados/Razões de Falha

Como é desde já possível de verificar, este CMS não apresenta nenhum problema de maior que impossibilite a sua utilização no Ci-FCUL no entanto convém referir alguns pontos respeitantes às principais vulnerabilidades deste gestor de conteúdos:

- O acesso aos diferentes conteúdos através de endereços URL próprios começou por ser um dos principais problemas do Joomla, uma vez que todas as páginas são acessíveis através da mesma morada, processando-se uma chamada interna da página de conteúdos que se quer visualizar (“query html strings”). O acesso externo a uma página específica não era por isso possível até ao lançamento de um módulo cujo objectivo é o de criação de aliáses para acesso às diferentes páginas de conteúdos embora estas conservem na realidade o mesmo endereço base.

- A ausência de um sistema de versionamento dos conteúdos no Joomla não representa uma falha relevante à sua perfeita utilização, uma vez que existe a possibilidade de os utilizadores pré-visualizarem um conteúdo antes de procederem à sua publicação, ou mesmo de o definirem como “rascunho” e só depois de uma atenta revisão, procederem a essa acção.

Drupal

O que é?

Fortemente marcado por um uso extensivo em todo o mundo, Drupal é uma plataforma de gestão de conteúdos Web de grande alcance que tanto cobre projectos simples como os projectos mais complexos.

A sua principal força e característica, é a forma como, facilmente, qualquer utilizador consegue criar novos conteúdos para um sítio Web como se estivesse simplesmente a navegar numa página. A juntar a esta facilidade, o Drupal conta ainda com o facto de ser o CMS que com mais facilidade estende as suas funcionalidades através de uma instalação facilitada de módulos e componentes.

Baseado em PHP, este CMS é eficiente sobre qualquer plataforma e compatível com quase qualquer sistema em que corra, sendo muito fácil de instalar, usar e gerir e totalmente configurável através da sua potente interface de administração.

Características principais para uso no CI

Criação e Edição de Conteúdo:

Drupal suporta originalmente dois tipos de conteúdos - Páginas e Histórias –, mas devido à sua grande flexibilidade, possibilita a criação de qualquer tipo de conteúdo por parte dos seus utilizadores, podendo estes escolher o número e tipo de campos que querem incluir no tipo de conteúdo que definem.

A criação de conteúdo é feita com base em HTML mas após a simples instalação de módulos básicos, torna-se possível a criação de conteúdo com o auxílio de editores simplificados (WYSIWYG) de texto, que permitem também a utilização simultânea de referências HTML e código PHP (como por exemplo a utilização de imagens agregadas ao texto) (Figura 75).

Existe uma função de pré-visualização do conteúdo no acto da sua criação, que permite verificar o resultado final evitando erros de conteúdo.

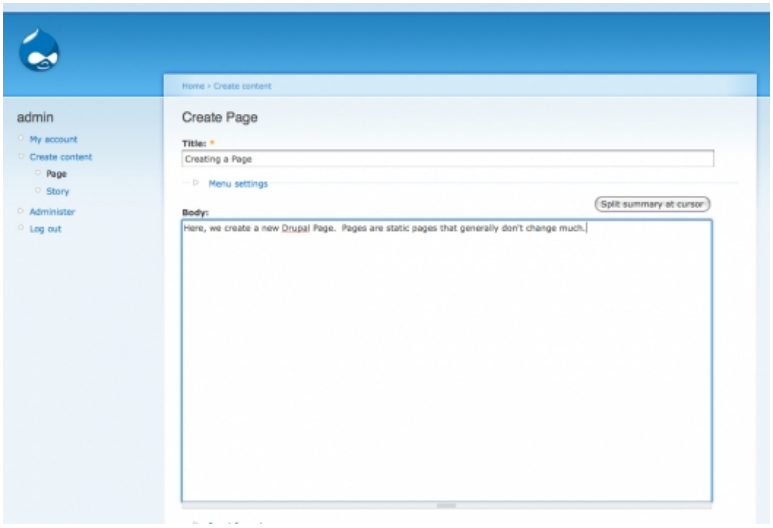


Figura 75: Criação de conteúdos no Drupal

Como já referi, através da instalação de simples módulos de funcionalidades, torna-se possível a criação de quase todos os tipos de conteúdos no Drupal. Desde painéis que reúnem diversos conteúdos, a livros de notícias, a menus personalizados e a separadores tipo Google. No Drupal quase tudo se torna possível.

Gestão:

Toda a gestão de um sítio Web gerido por Drupal é realizada através da própria página que serve de interface a qualquer tipo de operação (Figura 76). Desde a criação de conteúdos, à gestão destes, à instalação de módulos com novas funcionalidades, à configuração de templates de apresentação, à gestão de utilizadores e seus respectivos papéis na aplicação, ao acesso a relatórios de erros, á configuração dos tipos de navegação e menus de ligações do sítio. Tudo passa por uma interface irrealmente simples para conter tanta opção.

Qualquer novo módulo instalado passa também a ser acessível e configurável através desta interface e se tivermos em conta que é possível instalar neste CMS quase qualquer funcionalidade que se consiga imaginar, então está-se perto de dizer que é um Gestor de Conteúdos perfeito.

O ponto menos forte do Drupal será possivelmente a forma como a criação de diferentes sítios Web tem sempre de ser antecedida por operações de criação de ficheiros manualmente no servidor onde se encontra instalada a aplicação. Para que um sítio exista, é inclusivamente necessária, por vezes, alguma configuração ao nível do servidor Apache.

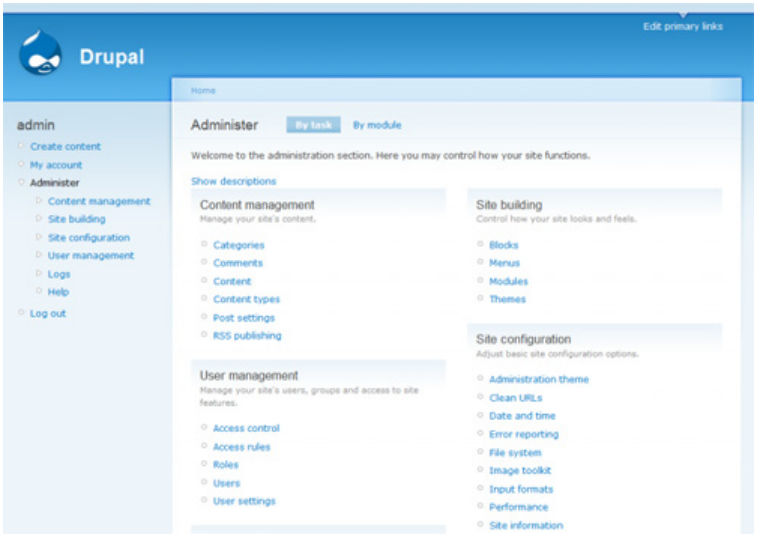


Figura 76: Interface de Administração do Drupal

Apresentação:

Ao nível da apresentação de conteúdos o Drupal CMS baseia-se na instalação e selecção de templates PHP com as suas próprias propriedades CSS e diferentes propriedades configuráveis.

Para criação de um template de apresentação é necessária muita programação nestas linguagens (PHP, CSS e XML) no entanto, o processo está bem definido no meio da comunidade Drupal, sabendo-se

especificamente quais as instruções e quais os tipos de ficheiros necessários para implementar um novo visual.

Através de diferentes módulos, este CMS vai para além da configuração de um template para apenas um sítio Web, sendo inclusivamente possível configurar diferentes apresentações para diferentes páginas, diferentes menus ou ainda diferentes tipos de conteúdos.

O acesso às diferentes páginas de conteúdos num sítio Web definido por esta ferramenta, permite também, por sua vez, uma integração posterior com outras plataformas, uma vez que para cada conteúdo criado (nó) é atribuída uma extensão de URL que o identifica no código HTML da página geral (são incluídos sobre forma de tags no conteúdo da página).

Comunidade e Suporte:

A comunidade Drupal é uma das suas principais forças e a razão de todo o seu sucesso e vasta utilização. Grande número de contribuições diárias são introduzidas via sítio Web e fórum oficial da aplicação todos os dias, sobre a forma tanto de módulos que implementam novas funcionalidades, como de novos templates de apresentação ou incrementos de segurança.

É também comum o pedido de novas funcionalidades por parte de utilizadores através do fórum, sendo dada grande resposta a estes pedidos, e sendo lançada, desta forma, por toda a comunidade uma onda de desenvolvimento de funcionalidades que são identificadas como estando em falta na plataforma.

Por fim, também a documentação está devidamente estruturada e organizada por uma serie de "livros"(handbooks) capazes de esclarecer qualquer dúvida ou guiar qualquer nova implementação ao nível deste software.

Problemas Identificados/Razões de Falha

Tal como o Joomla, o Drupal não apresenta razões de falha para que não se justifique a sua adopção por parte do CI-FCUL, sendo mesmo um dos principais candidatos a gestor de conteúdos Web, apresenta no entanto alguns pontos que podem ser interpretados como menos fortes, mas que de alguma forma são compensados pelas restantes funcionalidades do sistema:

- O facto de todo o template de apresentação ser programado apenas em PHP limita utilizadores menos experientes a uma definição total do seu conceito ideal de apresentação de componentes. No entanto, devido à grande quantidade de funcionalidades de personalização da apresentação, estes podem mesmo assim definir os conteúdos que querem mostrar e os locais da página em que querem mostrar esses conteúdos, definir imagens de fundo, imagens de cabeçalho, frases de cabeçalho, frases de manutenção, etc. Essa grande quantidade de traços personalizáveis compensa a rija programação do template inicial (tal como a grande quantidade de templates disponíveis gratuitamente para instalação também o compensa).

- No Drupal é possível configurar e criar qualquer número de sites e domínios que se queira, no entanto essa configuração é sempre feita mediante o cumprimento de dois ou três passos de criação e cópia de ficheiros na raiz da instalação. Esse tipo de instruções pode não estar ao alcance de utilizadores básicos

mas uma vez que, de qualquer forma, a configuração e instalação de todo o CMS nunca deverá estar entregue a um utilizador desse tipo, esse aspecto deixa de ter grande relevância.